

The l3backend-pdfmanagement package

Additional backend PDF features

L^AT_EX PDF management bundle

The L^AT_EX Project*

Version 0.97c, released 2026-05-26

1 l3backend-pdfmanagement Implementation

```
1 <drivers>\ProvidesExplFile
2 <*dvipdfmx>
3   {l3backend-pdf-dvipdfmx.def}{2026-05-26}{0.97c}
4   {LaTeX~PDF~management~bundle~backend~support:~dvipdfmx}
5 </dvipdfmx>
6 <*dvips>
7   {l3backend-pdf-dvips.def}{2026-05-26}{0.97c}
8   {LaTeX~PDF~management~bundle~backend~support:~dvips}
9 </dvips>
10 <*dvisvgm>
11   {l3backend-pdf-dvisvgm.def}{2026-05-26}{0.97c}
12   {LaTeX~PDF~management~bundle~backend~support:~dvisvgm}
13 </dvisvgm>
14 <*luatex>
15   {l3backend-pdf-luatex.def}{2026-05-26}{0.97c}
16   {LaTeX~PDF~management~bundle~backend~support:~PDF~output~(LuaTeX)}
17 </luatex>
18 <*pdftex>
19   {l3backend-pdf-pdftex.def}{2026-05-26}{0.97c}
20   {LaTeX~PDF~management~bundle~backend~support:~PDF~output~(pdfTeX)}
21 </pdftex>
22 <*xdvipdfmx>
23   {l3backend-pdf-xetex.def}{2026-05-26}{0.97c}
24   {LaTeX~PDF~management~bundle~backend~support:~XeTeX}
25 </xdvipdfmx>
```

1.1 Variants

We need to generate temporarily a few e-types variants of kernel backend commands. These can be removed once the kernel provides them.

```
26 <@@=pdf>
27 <*luatex | pdftex>
28 \cs_generate_variant:Nn \__kernel_backend_literal_page:n { e }
```

*E-mail: latex-team@latex-project.org

```

29 </luatex | pdftex>
30 < *dvipdfmx | xdvipdfmx>
31 \cs_generate_variant:Nn \__kernel_backend_literal:n { e }
32 \cs_generate_variant:Nn \__pdf_backend:n { e }
33 </dvipdfmx | xdvipdfmx>
34 < *dvips>
35 \cs_generate_variant:Nn \__kernel_backend_postscript:n { e }
36 \cs_generate_variant:Nn \__pdf_backend_pdfmark:n { e }
37 </dvips>

```

1.2 Support for delayed literal and special

Starting with TeXlive 2023 the engines support a `shipout` keyword for `\pdfliteral` and `\special`. When used the argument is not expanded when the command is used but only when the page is shipped out. This allows for example the tagging code to delay the page-wise numbering of MC-chunks until the page is actually built. For now we test the engine support. The boolean is setup in `pdfmanagement-testphase.dtx`.

```

38 < *drivers>

```

The following commands provide the needed kernel backend support. This are basically copies of similar commands of `l3backend-basics`.

`__kernel_backend_shipout_literal:e` The one shared function for all backends is access to the basic `\special` primitive.

```

39 \cs_new_protected:Npn \__kernel_backend_shipout_literal:e #1
40 { \tex_special:D~shipout { #1 } }
41 </drivers>

```

(End of definition for __kernel_backend_shipout_literal:e.)

```

42 < *luatex | pdftex>

```

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

```

43 \cs_new_protected:Npn \__kernel_backend_shipout_literal_pdf:e #1
44 {
45 < *luatex>
46   \tex_pdfextension:D ~ literal ~ shipout ~
47 </luatex>
48 < *pdftex>
49   \tex_pdfliteral:D ~ shipout ~
50 </pdftex>
51   { #1 }
52 }

```

(End of definition for __kernel_backend_shipout_literal_pdf:e.)

`__kernel_backend_shipout_literal_page:e` Page literals are pretty simple.

```

53 \cs_new_protected:Npn \__kernel_backend_shipout_literal_page:e #1
54 {
55 < *luatex>
56   \tex_pdfextension:D ~ literal ~ shipout ~
57 </luatex>
58 < *pdftex>
59   \tex_pdfliteral:D ~ shipout ~

```

```

60 </pdfTeX>
61     page { #1 }
62 }
63 </luatex | pdfTeX>

```

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

1.3 Crossreferences

Commands to get a reference for the absolute page counter.

```

64 <*drivers>
65 \cs_new_protected:Npn \_pdf_backend_record_abspage:n #1
66 {
67     \@bsphack
68     \property_record:nn{#1}{abspage}
69     \@esphack
70 }
71 \cs_new:Npn \_pdf_backend_ref_abspage:n #1
72 {
73     \property_ref:nn{#1}{abspage}
74 }
75
76 \cs_generate_variant:Nn \_pdf_backend_record_abspage:n {e}
77 \cs_generate_variant:Nn \_pdf_backend_ref_abspage:n {e}
78 </drivers>

```

avoid that destinations names are optimized with xelatex/dvipdfmx see <https://tug.org/piper-mail/dvipdfmx/2019-May/000002.html>

```

79 <dvipdfmx | xdvipdfmx>
80     \_kernel_backend_literal:n { dvipdfmx:config-C~ 0x0010 }
81 </dvipdfmx | xdvipdfmx>

```

```

\g__pdf_tmpa_prop
\l__pdf_tmpa_tl
\l__pdf_backend_tmpa_box

```

Some scratch variables

```

82 <*drivers>
83 \prop_new:N \g__pdf_tmpa_prop
84 \tl_new:N \l__pdf_tmpa_tl
85 \box_new:N \l__pdf_backend_tmpa_box
86 \box_new:N \l__pdf_backend_tmppb_box
87 </drivers>

```

(End of definition for `\g__pdf_tmpa_prop`, `\l__pdf_tmpa_tl`, and `\l__pdf_backend_tmpa_box`.)

```

\g__pdf_backend_resourceid_int
\g__pdf_backend_name_int
\g__pdf_backend_page_int

```

a counter to create labels for the resources, a counter to number properties in bdc marks, a counter for the `\pdfpageref` implementation.

```

88 <*drivers>
89 \int_new:N \g__pdf_backend_resourceid_int
90 \int_new:N \g__pdf_backend_name_int
91 \int_new:N \g__pdf_backend_page_int
92 </drivers>

```

(End of definition for `\g__pdf_backend_resourceid_int`, `\g__pdf_backend_name_int`, and `\g__pdf_backend_page_int`.)

1.4 luacode

Load the lua code.

```
93 <*luatex>
94     \directlua { require("l3backend-pdf.lua") }
95 </luatex>
```

1.5 Converting unicode strings to a pdfname

dvips needs a special function here, so we add this as backend function.

```
96 <*pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
97 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
98 {
99     / \str_convert_pdfname:e { \text_expand:n { #1 } }
100 }
101 </pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
102 <*dvips>
103 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
104 {
105     ~ ( \text_expand:n { #1 } ) ~ cvn
106 }
107 </dvips>
```

1.6 Hooks

1.6.1 Add the “end run” hooks

Here we add the end run hook to suitable end hooks.

```
108 <*pdftex | luatex>
109 % put in \@kernel@after@enddocument@afterlastpage
110 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
111 {
112     \g__kernel_pdfmanagement_end_run_code_tl
113 }
114 </pdftex | luatex>
115 <*dvipdfmx | xdvipdfmx>
116 % put in \@kernel@after@shipout@lastpage
117 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
118 {
119     \g__kernel_pdfmanagement_end_run_code_tl
120 }
121 </dvipdfmx | xdvipdfmx>
122 <*dvips>
123 % put in \@kernel@after@shipout@lastpage
124 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
125 {
126     \g__kernel_pdfmanagement_end_run_code_tl
127 }
128 </dvips>
```

1.6.2 Add the “shipout” hooks

Now we add to the shipout hooks the relevant token lists. We also push the page resources in shipout/firstpage (AtBeginDvi) as the backend code sets color stack there. The xetex driver needs a rule here. If it clashes on the first page, we will need a test ...

```

129 <*drivers>
130 \tl_if_exist:NTF \@kernel@after@shipout@background
131 {
132   \g@addto@macro \@kernel@before@shipout@background{\relax}
133   \g@addto@macro \@kernel@after@shipout@background
134   {
135     \g__kernel_pdfmanagement_thispage_shipout_code_tl
136   }
137 }
138 {
139   \hook_gput_code:nnn{shipout/background}{pdf}
140   {
141     \g__kernel_pdfmanagement_thispage_shipout_code_tl
142   }
143 }
144
145 </drivers>

```

1.7 The /Pages dictionary (pdfpagesattr)

_pdf_backend_Pages_primitive:n

This is the primitive command to add something to the /Pages dictionary. It works differently for the backends: pdftex and luatex overwrite existing content, dvips and dvipdfmx are additive. luatex sets it in lua. The higher level code has to take this into account.

```

146 <*pdftex>
147 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
148 {
149   \tex_global:D \tex_pdfpagesattr:D { #1 }
150 }
151 </pdftex>
152 <*luatex>
153 %luatex: does it in lua
154 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
155 {
156   \tex_directlua:D
157   {
158     pdf.setpagesattributes( \_pdf_backend_luastring:n { #1 } )
159   }
160 }
161 </luatex>
162 <*dvips>
163 \cs_new_protected:Npx \_pdf_backend_Pages_primitive:n #1
164 {
165   \tex_special:D{ps:~[#1~/PAGES~pdfmark} %]
166 }
167 </dvips>
168 <*dvipdfmx | xdvipdfmx>
169 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1

```

```

170 {
171   \__pdf_backend:n{put~@pages~<<#1>>}
172 }
173 </dvipdfmx | xdvipdfmx>
174 <*dvisvgm>
175 \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
176 {}
177 </dvisvgm>

```

(End of definition for __pdf_backend_Pages_primitive:n.)

1.8 “Page” and “ThisPage” attributes (pdfpageattr)

__pdf_backend_Page_primitive:n is the primitive command to add something to the /Page dictionary. It works differently for the backends: pdftex and luatex overwrite existing content, dvips and dvipdfmx are additive. luatex sets it in lua. The higher level code has to take this into account. __pdf_backend_Page_gput:nn stores default values. __pdf_backend_Page_gremove:n allows to remove a value. __pdf_backend_ThisPage_gput:nn adds a value to the current page. __pdf_backend_ThisPage_gpush:n merges the default and the current page values and add them to the dictionary of the current page in \g__pdf_backend_thispage_shipout_tl.

```

178 % backend commands
179 <*pdftex>
180 %the primitive
181 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
182 {
183   \tex_global:D \tex_pdfpageattr:D { #1 }
184 }
185 % the command to store default values.
186 % Uses a prop with pdflatex + dvi,
187 % sets a lua table with luatex
188 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2 %key,value
189 {
190   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
191 }
192 % the command to remove a default value.
193 % Uses a prop with pdflatex + dvi,
194 % changes a lua table with luatex
195 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
196 {
197   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
198 }
199 % the command used in the document.
200 % direct call of the primitive special with dvips/dvipdfmx
201 % \latelua: fill a page related table with luatex, merge it with the page
202 % table and push it directly
203 % write to aux and store in prop with pdflatex
204 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
205 {
206   %we need to know the page the resource should be added too.
207   \int_gincr:N\g__pdf_backend_resourceid_int
208   \__pdf_backend_record_abspace:e { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
209   \tl_set:Ne \l__pdf_tmpa_tl

```

```

210     {
211       \__pdf_backend_ref_abspage:e {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
212     }
213     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
214     {
215       \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
216     }
217     %backend_Page has no handler.
218     \pdfdict_gput:nnn {g__pdf_Core/backend_Page\l__pdf_tmpa_tl}{ #1 }{ #2 }
219   }
220   %the code to push the values, used in shipout
221   %merges the two props and then fills the register in pdflatex
222   %merges the two tables and then fills (in lua) in luatex
223   %issues the values stored in the global prop with dvi
224   \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
225   {
226     \prop_gset_eq:Nc \g__pdf_tmpa_prop { \__kernel_pdfdict_name:n { g__pdf_Core/Page } }
227     \prop_if_exist:cT { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
228     {
229       \prop_map_inline:cn { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
230       {
231         \prop_gput:Nnn \g__pdf_tmpa_prop { ##1 }{ ##2 }
232       }
233     }
234     \__pdf_backend_Page_primitive:e
235     {
236       \prop_map_function:NN \g__pdf_tmpa_prop \pdfdict_item:ne
237     }
238   }
239   </pdftex>
240   <*luatex>
241   % do we need to use some escaping for the values????
242   \cs_new:Npn \__pdf_backend_luastring:n #1
243   {
244     "\tex_luaescapestring:D { \tex_unexpanded:D { #1 } }"
245   }
246   %not used, only there for consistency
247   \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
248   {
249     \tex_latelua:D
250     {
251       pdf.setpageattributes(\__pdf_backend_luastring:n { #1 })
252     }
253   }
254   % the command to store default values.
255   % Uses a prop with pdflatex + dvi,
256   % sets a lua table with luatex
257   \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
258   {
259     \tex_directlua:D
260     {
261       ltx.__pdf.backend_Page_gput
262       (
263         \__pdf_backend_luastring:n { #1 },

```

```

264         \_pdf_backend_luastring:n { #2 }
265     )
266 }
267 }
268 % the command to remove a default value.
269 % Uses a prop with pdflatex + dvi,
270 % changes a lua table with luatex
271 \cs_new_protected:Npn \_pdf_backend_Page_gremove:n #1
272 {
273     \tex_directlua:D
274     {
275         ltx._pdf.backend_Page_gremove (\_pdf_backend_luastring:n { #1 })
276     }
277 }
278 % the command used in the document.
279 % direct call of the primitive special with dvips/dvipdfmx
280 % \latelua: fill a page related table with luatex, merge it with the page
281 % table and push it directly
282 % write to aux and store in prop with pdflatex
283 \cs_new_protected:Npn \_pdf_backend_ThisPage_gput:nn #1 #2
284 {
285     \tex_latelua:D
286     {
287         ltx._pdf.backend_ThisPage_gput
288         (
289             tex.count["g_shipout_readonly_int"],
290             \_pdf_backend_luastring:n { #1 },
291             \_pdf_backend_luastring:n { #2 }
292         )
293         ltx._pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
294     }
295 }
296 %the code to push the values, used in shipout
297 %merges the two props and then fills the register in pdflatex
298 %merges the two tables (the one is probably still empty) and then fills (in lua) in luatex
299 %issues the values stored in the global prop with dvi
300 \cs_new_protected:Npn \_pdf_backend_ThisPage_gpush:n #1
301 {
302     \tex_latelua:D
303     {
304         ltx._pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
305     }
306 }
307
308 </luatex>
309 <*dvipdfmx | xdvipdfmx>
310 %the primitive
311 \cs_new_protected:Npn \_pdf_backend_Page_primitive:n #1
312 {
313     \tex_special:D{pdf:-put~@thispage-<<#1>>}
314 }
315 % the command to store default values.
316 % Uses a prop with pdflatex + dvi,
317 % sets a lua table with luatex

```



```

318 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
319 {
320   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
321 }
322 % the command to remove a default value.
323 % Uses a prop with pdflatex + dvi,
324 % changes a lua table with luatex
325 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
326 {
327   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
328 }
329 % the command used in the document.
330 % direct call of the primitive special with dvips/dvipdfmx
331 % \latelua: fill a page related table with luatex, merge it with the page
332 % table and push it directly
333 % write to aux and store in prop with pdflatex
334 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
335 {
336   \__pdf_backend_Page_primitive:n { /#1~#2 }
337 }
338 %the code to push the values, used in shipout
339 %merges the two props and then fills the register in pdflatex
340 %merges the two tables (the one is probably still empty)
341 % and then fills (in lua) in luatex
342 %issues the values stored in the global prop with dvi
343 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
344 {
345   \__pdf_backend_Page_primitive:e
346   { \pdfdict_use:n { g__pdf_Core/Page} }
347 }
348 </dvipdfmx | xdvipdfmx>
349 <*dvips>
350 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
351 {
352   \tex_special:D{ps:-[ThisPage]<<#1>>~/PUT~pdfmark} %]
353 }
354 % the command to store default values.
355 % Uses a prop with pdflatex + dvi,
356 % sets a lua table with luatex
357 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
358 {
359   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
360 }
361 % the command to remove a default value.
362 % Uses a prop with pdflatex + dvi,
363 % changes a lua table with luatex
364 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
365 {
366   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
367 }
368 % the command used in the document.
369 % direct call of the primitive special with dvips/dvipdfmx
370 % \latelua: fill a page related table with luatex, merge it with the page
371 % table and push it directly

```

```

372 % write to aux and store in prop with pdflatex
373 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
374 {
375   \__pdf_backend_Page_primitive:n { /#1~#2 }
376 }
377 %the code to push the values, used in shipout
378 %merges the two props and then fills the register in pdflatex
379 %merges the two tables (the one is probably still empty)
380 %and then fills (in lua) in luatex
381 %issues the values stored in the global prop with dvi
382 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
383 {
384   \__pdf_backend_Page_primitive:e
385   { \pdfdict_use:n { g__pdf_Core/Page} }
386 }
387 </dvips>
388 <*dvisvgm>
389 % mostly only dummies ...
390 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
391 {}
392 % Uses a prop with pdflatex + dvi,
393 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
394 {
395   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
396 }
397 % the command to remove a default value.
398 % Uses a prop with pdflatex + dvi,
399 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
400 {
401   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
402 }
403 % the command used in the document.
404 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
405 {}
406 %the code to push the values, used in shipout
407 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
408 {}
409 </dvisvgm>
410 <*drivers>
411 \cs_generate_variant:Nn \__pdf_backend_Page_primitive:n { e }
412 </drivers>

```

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

1.9 “Page/Resources”: ExtGState, ColorSpace, Shading, Pattern

Path: Page/Resources/ExtGState etc. The actual output of the resources is handled together with the bdc/Properties. Here is only special code.

\c__pdf_backend_PageResources_clist

The names are quite often needed a similar list is now in l3pdfmanagement. Perhaps it should be merged.

```

413 <*drivers>
414 \clist_const:Nn \c__pdf_backend_PageResources_clist

```

```

415 {
416   ExtGState,
417   ColorSpace,
418   Pattern,
419   Shading,
420 }
421 </drivers>

```

(End of definition for `\c__pdf_backend_PageResources_clist`.)

Now the backend commands the command to fill the register and to push the values.

`__pdf_backend_PageResources_gput:nnn` stores values for the page resources.

#1 : name of the resource (ExtGState, ColorSpace, Shading, Pattern)
 #2 : a pdf name without slash
 #3 : value

This pushes out the objects. It should be a no-op with xdvipdfmx and dvips as it is currently issued in the end-of-run hook! create the backend objects:

`__pdf_backend_PageResources_obj_gpush:`

```

422 <*pdfTeX | luatex>
423 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
424 {
425   \pdf_object_new:n {\__pdf/Page/Resources/#1}
426   \cs_if_exist:NT \tex_directlua:D
427   {
428     \tex_directlua:D
429     {
430       ltx.__pdf.object["__pdf/Page/Resources/#1"]
431       =
432       "\pdf_object_ref:n{\__pdf/Page/Resources/#1}"
433     }
434   }
435 }
436 </pdfTeX | luatex>

```

values are only stored in a prop and will be output at end document. luatex must also trigger the lua side

```

437 <*luatex>
438 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
439 {
440   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
441   \tex_directlua:D{ltx.__pdf.Page.Resources.#1=true}
442   \tex_directlua:D
443   {
444     ltx.pdf.Page.Resources_gpush(tex.count["g_shipout_readonly_int"])
445   }
446 }
447 </luatex>
448 <*pdfTeX>
449 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
450 {
451   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
452 }
453 </pdfTeX>

```

code for end of document code. Change 2026-03-11: we push out the objects even if they are empty as tikz references them in X-objects (fadings)

```

454 <*pdftex | luatex>
455 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush:
456 {
457   \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
458   {
459     \pdf_object_write:nne
460     { __pdf/Page/Resources/##1 } { dict }
461     { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1 } }
462   }
463 }
464 </pdftex | luatex>

```

xdvipdfmx doesn't work correctly with object names ... <https://tug.org/piper-mail/dvipdfmx/2019-August/000021.html>, so we use this must be issued on every page! objects should not only be created but also initialized initialization should be done before anyone tries to write so we add rules for the backend. The push command should not be used as it is in the wrong end document hook. If needed a new command must be added.

```

465 <*dvipdfmx | xdvipdfmx>
466 <xdvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-xetex}{after}{pdf}
467 <dvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-dvipdfmx}{after}{pdf}
468 %
469 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
470 {
471   \pdf_object_new:n { __pdf/Page/Resources/#1 }
472   \hook_gput_code:nnn
473   {shipout/firstpage}
474   {pdf}
475   {\pdf_object_write:nnn { __pdf/Page/Resources/#1 } { dict } {}}
476 }
477 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1
478 {
479   \__pdf_backend:n {put~@resources~<<#1>>}
480 }
481 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
482 {
483   % this is not used for output, but there is a test if the resource is empty
484   \prop_gput:cne { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1 } }
485   { \str_convert_pdfname:n {#2} }{ #3 }
486   %objects are not filled with \pdf_object_write as this is not additive!
487   \__pdf_backend:e
488   {
489     put~\pdf_object_ref:n {__pdf/Page/Resources/#1}<</#2~#3>>
490   }
491 }
492
493 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
494 </dvipdfmx | xdvipdfmx>

```

dvips unneeded, or no-op. The push command should not be used as it is in the wrong end document hook. If needed a new command must be added.

```

495 <*dvips>

```

```

496 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
497 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
498 { %only for the show command TEST!!
499   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
500 }
501 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
502 </dvips>

```

dvipsvgm unneeded, or no-op

```

503 <*dvipsvgm>
504 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
505 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
506 { %only for the show command TEST!!
507   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
508 }
509 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
510 </dvipsvgm>

```

(End of definition for __pdf_backend_PageResources_gput:nnn and __pdf_backend_PageResources_obj_gpush:.)

1.9.1 Page resources /Properties + BDC operators

__pdf_backend_bdc:nn __pdf_backend_bdc:nn, __pdf_backend_shipout_bdc:ee, __pdf_backend_bdcobject:nn, __pdf_backend_bdcobject:n, __pdf_backend_bmc:n and __pdf_backend_emc: are the backend command that create the bdc/emc marker and store the properties. __pdf_backend_bdcobject:n __pdf_backend_PageResources_gpush:n outputs the /Properties and/or the other resources for the current page.

pdftex and luatex (and perhaps dvips ...) need to know if there are in a xform stream ...

```

511 <*drivers>
512 \bool_new:N \l__pdf_backend_xform_bool
513 </drivers>

```

dvips is easy: create an object, and reference it in the bdc ghostscript will then automatically replace it by a name and add the name to the /Properties dict, special variant von accsupp <https://chat.stackexchange.com/transcript/message/50831812#50831812>

```

514 <*dvips>
515 %
516 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
517 {
518   \__pdf_backend_pdfmark:n{/#1~<<#2>>~/BDC}
519 }

```

There is not difference here between inline and property BDC, it is always a property:

```

520 \cs_set_eq:NN \__pdf_backend_bdc_contobj:nn \__pdf_backend_bdc:nn
521 \cs_set_eq:NN \__pdf_backend_bdc_contstream:nn \__pdf_backend_bdc:nn
522
523 \cs_new_protected:Npn \__pdf_backend_bdc_shipout:ee #1 #2 % #1 eg. Span, #2: dict_content
524 {
525   \__kernel_backend_shipout_literal:e
526   {ps: SDict ~ begin ~ mark /#1~<<#2>>~/BDC ~ pdfmark ~ end }
527 }
528

```

```

529 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
530 {
531   \__pdf_backend_pdfmark:e{/#1~\pdf_object_ref:n{#2}~/BDC}
532 }
533 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
534 {
535   \__pdf_backend_pdfmark:e{/#1~\__pdf_backend_object_last:~/BDC}
536 }
537 \cs_set_protected:Npn \__pdf_backend_emc:
538 {
539   \__pdf_backend_pdfmark:n{/EMC} %
540 }
541 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
542 {
543   \__pdf_backend_pdfmark:n{/#1~/BMC} %
544 }
545 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
546
547 </dvips>
548 <*dvisvgm>
549 % dvisvgm should do nothing
550 %
551 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
552 {}
553 \cs_set_eq:NN \__pdf_backend_bdc_contobj:nn \__pdf_backend_bdc:nn
554 \cs_set_eq:NN \__pdf_backend_bdc_contstream:nn \__pdf_backend_bdc:nn
555
556 \cs_set_protected:Npn \__pdf_backend_shipout_bdc:ee #1 #2 % #1 eg. Span, #2: dict_content
557 {}
558 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
559 {}
560 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
561 {}
562 \cs_set_protected:Npn \__pdf_backend_emc:
563 {}
564 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
565 {}
566 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
567
568 </dvisvgm>
569 %
570 % xetex has to create the entries in the /Properties manually
571 % (like the other backends)
572 % use pdfbase special
573 % https://chat.stackexchange.com/transcript/message/50832016#50832016
574 % the property is added to xform resources automatically,
575 % no need to worry about it.
576 <*dvipdfmx|xdvipdfmx>
577 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
578 {
579   \int_gincr:N \g__pdf_backend_name_int
580   \__kernel_backend_literal:e
581   {
582     pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC

```

```

583     }
584     \__kernel_backend_literal:e
585     {
586         pdf:put~@resources~
587         <<
588         /Properties~
589         <<
590         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_t1
591         \pdf_object_ref:n { #2 }
592         >>
593     >>
594     }
595 }
596 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span
597 {
598     \int_gincr:N \g__pdf_backend_name_int
599     \__kernel_backend_literal:e
600     {
601         pdf:code~/\exp_not:n{#1}/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_t1 BDC
602     }
603     \__kernel_backend_literal:e
604     {
605         pdf:put~@resources~
606         <<
607         /Properties~
608         <<
609         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_t1
610         \__pdf_backend_object_last:
611         >>
612     >>
613     }
614 }
615 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
616 {
617     \__kernel_backend_literal:n {pdf:code~/#1~BMC} %pdfbase
618 }
619
620 %this require management
621 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
622 {
623     \pdf_object_unnamed_write:nn { dict }{ #2 }
624     \__pdf_backend_bdcobject:n { #1 }
625 }
626
627 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
628 {
629     \__kernel_backend_literal:n {pdf:code~ /#1~<<#2>>~BDC }
630 }
631
632 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
633 {
634     \cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn
635     \__pdf_backend_bdc:nn {#1}{#2}
636 }

```

```

637
638 \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
639 {
640   \__kernel_backend_shipout_literal:e {pdf:code~ /#1~<<#2>>~BDC }
641 }
642 \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
643
644 \cs_set_protected:Npn \__pdf_backend_emc:
645 {
646   \__kernel_backend_literal:n {pdf:code~EMC} %pdfbase
647 }
648 % properties are handled automatically, but the other resources should be added
649 % at shipout
650 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
651 {
652   \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
653   {
654     \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
655     {
656       \__kernel_backend_literal:e
657       {
658         pdf:put~@resources~
659         <</##1~\pdf_object_ref:n {__pdf/Page/Resources/##1}>>
660       }
661     }
662   }
663 }
664 </dvipdfmx|xdvipdfmx>
665 % luatex + pdftex
666 <*luatex>
667 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
668 {
669   \int_gincr:N \g__pdf_backend_name_int
670   \__kernel_backend_literal_page:e
671   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
672   \bool_if:NTF \l__pdf_backend_xform_bool
673   {
674     \pdfdict_gput:nee
675     { g__pdf_Core/Xform/Resources/Properties }
676     { l3pdf\int_use:N\g__pdf_backend_name_int }
677     { \pdf_object_ref:n { #2 } }
678   }
679   {
680     \exp_args:Ne \tex_latelua:D
681     {
682       ltx.pdf.Page_Resources_Properties_gput
683       (
684         tex.count["g_shipout_readonly_int"],
685         "l3pdf\int_use:N\g__pdf_backend_name_int",
686         "\pdf_object_ref:n { #2 }"
687       )
688     }
689   }
690 }

```



```

691 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
692 {
693   \int_gincr:N \g__pdf_backend_name_int
694   \__kernel_backend_literal_page:e
695   { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
696   \bool_if:NTF \l__pdf_backend_xform_bool
697   {
698     \pdfdict_gput:nee %no handler needed
699     { g__pdf_Core/Xform/Resources/Properties }
700     { l3pdf\int_use:N\g__pdf_backend_name_int }
701     { \__pdf_backend_object_last: }
702   }
703   {
704     \exp_args:Ne \tex_latelua:D
705     {
706       ltx.pdf.Page_Resources_Properties_gput
707       (
708         tex.count["g_shipout_readonly_int"],
709         "l3pdf\int_use:N\g__pdf_backend_name_int",
710         "\__pdf_backend_object_last:"
711       )
712     }
713   }
714 }
715 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
716 {
717   \__kernel_backend_literal_page:n { /#1~BMC }
718 }
719 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
720 {
721   \pdf_object_unnamed_write:nn { dict } { #2 }
722   \__pdf_backend_bdcobject:n { #1 }
723 }
724 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
725 {
726   \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
727 }
728
729 \cs_set_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn
730
731 \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
732 {
733   \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
734 }
735 \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
736
737 \cs_set_protected:Npn \__pdf_backend_emc:
738 {
739   \__kernel_backend_literal_page:n { EMC }
740 }
741
742 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
743 </luatex>

```

pdf_latex is the most complicated if we want to use properties as it has to go through the aux ... the push command is extended to take other resources too

```

744 <*pdftex>
745 \csset_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
746 {
747   \int_gincr:N \g__pdf_backend_name_int
748   \__kernel_backend_literal_page:e
749   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
750   % code to set the property ....
751   \int_gincr:N\g__pdf_backend_resourceid_int
752   \bool_if:NTF \l__pdf_backend_xform_bool
753   {
754     \pdfdict_gput:nee %no handler needed
755     { g__pdf_Core/Xform/Resources/Properties }
756     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
757     { \pdf_object_ref:n { #2 } }
758   }
759   {
760     \__pdf_backend_record_abspage:e {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
761     \tl_set:Ne \l__pdf_tmpa_tl
762     {
763       \__pdf_backend_ref_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
764     }
765     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
766     {
767       \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
768     }
769     \pdfdict_gput:nee
770     { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
771     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
772     { \pdf_object_ref:n{#2} }
773   }
774 }
775 \csset_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
776 {
777   \int_gincr:N \g__pdf_backend_name_int
778   \__kernel_backend_literal_page:e
779   { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
780   % code to set the property ....
781   \int_gincr:N\g__pdf_backend_resourceid_int
782   \bool_if:NTF \l__pdf_backend_xform_bool
783   {
784     \pdfdict_gput:nee
785     { g__pdf_Core/Xform/Resources/Properties }
786     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
787     { \__pdf_backend_object_last: }
788   }
789   {
790     \__pdf_backend_record_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
791     \tl_set:Ne \l__pdf_tmpa_tl
792     {
793       \__pdf_backend_ref_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
794     }
795     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }

```

```

796         {
797             \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
798         }
799         \pdfdict_gput:nee
800         { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
801         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
802         { \__pdf_backend_object_last: }
803         %\pdfdict_show:n { g_backend_Page\l__pdf_tmpa_tl/Resources/Properties }
804     }
805 }
806 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
807 {
808     \__kernel_backend_literal_page:n { /#1~BMC }
809 }
810 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
811 {
812     \pdf_object_unnamed_write:nn { dict } { #2 }
813     \__pdf_backend_bdcobject:n { #1 }
814 }
815 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
816 {
817     \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
818 }

```

We use by default the direct BDC.

```

819 \cs_set_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn
820
821 \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
822 {
823     \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
824 }
825 \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
826
827
828 \cs_set_protected:Npn \__pdf_backend_emc:
829 {
830     \__kernel_backend_literal_page:n { EMC }
831 }
832
833 \cs_new:Npn \__pdf_backend_PageResources_gpush_aux:n #1 %#1 ExtGState etc
834 {
835     \prop_if_empty:cF
836     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1 } }
837     {
838         \pdfdict_item:ne { #1 } { \pdf_object_ref:n { __pdf/Page/Resources/#1 } }
839     }
840 }
841
842 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
843 {
844     \exp_args:NNe \tex_global:D \tex_pdfpageresources:D
845     {
846         \prop_if_exist:cT
847         { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
848         {

```

```

849         /Properties~
850         <<
851         \prop_map_function:cN
852         { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties
853         \pdfdict_item:ne
854         >>
855     }
856     %% add ExtGState etc
857     \clist_map_function:NN
858     \c__pdf_backend_PageResources_clist
859     \__pdf_backend_PageResources_gpunch_aux:n
860 }
861 }
862
863 </pdftex>

```

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

1.10 “Catalog” & subdirectories (pdfcatalog)

The backend command is already in the driver: __pdf_backend_catalog_gput:nn

1.10.1 Special case: the /Names/EmbeddedFiles dictionary

Entries to /Names are handled differently, in part (/Desc) it is automatic, for other special commands like \pdfnames must be used. For EmbeddedFiles dvips wants code for every file and then creates the Name tree automatically. Other name trees are ignored. TODO: Currently the code for EmbeddedFiles is still a bit different but this should be merged, all name trees should be handled with the same code.

```

864 % pdflatex
865 <*pdftex>
866 \cs_new_protected:Npn \__pdf_backend_Names_gpunch:nn #1 #2 %#1 name of name tree, #2 array co
867 {
868     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
869     \tex_pdfnames:D {/#1~\pdf_object_ref_last:}
870 }
871 </pdftex>
872 <*luatex>
873 \cs_new_protected:Npn \__pdf_backend_Names_gpunch:nn #1 #2 %#1 name of name tree, #2 array co
874 {
875     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
876     \tex_pdfextension:D~names~ {/#1~\pdf_object_ref_last:}
877 }
878 </luatex>
879 <*dviPDFmx | xdvipdfmx>
880 \cs_new_protected:Npn \__pdf_backend_Names_gpunch:nn #1 #2 %#1 name of name tree, #2 array co
881 {
882     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
883     \__pdf_backend:e {put~@names~</#1~\pdf_object_ref_last: >>}
884 }
885 </dviPDFmx | xdvipdfmx>
886
887 %dvips: noop
888 <*dvips>

```

```

889 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
890 </dvips>
891 %dvisvgm: noop
892 <*dvisvgm>
893 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
894 </dvisvgm>

```

EmbeddedFiles is a bit special. For once we need backend commands for dvips. But we want also an option to create the name on the fly.

__pdf_backend_NamesEmbeddedFiles_add:nn

dvips need special backend code to create the name tree. With the other engines it does nothing.

```

895 <*pdftex | luatex | dvipdfmx | xdvipdfmx>
896 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2 {}
897 </pdftex | luatex | dvipdfmx | xdvipdfmx>
898 <*dvips>
899 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
900 {
901     \__pdf_backend_pdfmark:e
902     {
903         /Name~#1~
904         /FS~#2~
905         /EMBED
906     }
907 }
908 </dvips>
909 <*dvisvgm>
910 %no op. Or is there any sensible use for it?
911 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
912 {}
913
914 </dvisvgm>

```

(End of definition for __pdf_backend_NamesEmbeddedFiles_add:nn.)

1.10.2 Additional annotation commands

Starting with texlive 2021 pdftex and luatex offer commands to interrupt a link. That can for example be used to exclude the header and footer from the link. The backend support is now in l3kernel. We only provide the user command.

```

915 <*pdftex>
916 \cs_if_exist:NT \pdfrunninglinkoff
917 {
918     \cs_set_protected:Npn \__pdfannot_backend_link_off:
919     {
920         \pdfrunninglinkoff
921     }
922     \cs_set_protected:Npn \__pdfannot_backend_link_on:
923     {
924         \pdfrunninglinkon
925     }
926 }
927 </pdftex>

```

1.10.3 Split links

With luatex we use luacode to handle link annotations. This allows us to retrieve the annotations as needed by the tagging code. It also allow to prevent that link areas spill over into unwanted regions like footnotes. See the documentation in lualinksplit.lua for more details. We therefore add a plug for the build/column/footnotes.

```

928 \*luatex
929 \lua_load_module:n{lualinksplit}
930 \NewSocketPlug{build/column/footnotes}{lualinksplit}{%
931   \setbox\footins=\vbox{\pdfextension linkstate-2\unvbox\footins}%
932 }
933 \AssignSocketPlug{build/column/footnotes}{lualinksplit}
934 \</luatex>

```

1.10.4 FormXObject / backend

```

\__pdf_backend_xform_new:nnnn #1 : name
                              #2 : attributes
                              #3 : resources needed?? or are all resources autogenerated?
                              #4 : content, this doesn't need to be a box!

```

```

\__pdf_backend_xform_use:n
\__pdf_backend_xform_ref:n
935 \*pdfTeX
936 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
937 % #1 name
938 % #2 attributes
939 % #3 resources
940 % #4 content, not necessarily a box!
941 {
942   \hbox_set:Nn \l__pdf_backend_tmpa_box
943   {
944     \bool_set_true:N \l__pdf_backend_xform_bool
945     \prop_gc_clear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
946     #4
947   }
948   %store the dimensions
949   \tl_const:ce
950   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
951   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
952   \tl_const:ce
953   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
954   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
955   \tl_const:ce
956   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
957   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
958   %% do we need to test if #2 and #3 are empty??
959   \tex_immediate:D \tex_pdfxform:D
960   ~ attr ~ { #2 }
961   %% which other resources should be default? Is an argument actually needed?
962   ~ resources ~
963   {
964     #3
965     \int_compare:nNnT

```

```

966         { \prop_count:c { \__kernel_pdffdict_name:n { g__pdf_Core/Xform/Resources/Properties
967         >
968         { 0 }
969         {
970         /Properties~
971         <<
972         \pdffdict_use:n { g__pdf_Core/Xform/Resources/Properties }
973         >>
974         }
975
976     \prop_if_empty:cF
977     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
978     {
979     /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
980     }
981     \prop_if_empty:cF
982     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
983     {
984     /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
985     }
986     \prop_if_empty:cF
987     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/Shading } }
988     {
989     /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
990     }
991     \prop_if_empty:cF
992     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
993     {
994     /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
995     }
996     }
997     \l__pdf_backend_tmpa_box
998     \int_const:cn
999     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1000    { \tex_pdflastxform:D }
1001    }
1002
1003    \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1004    {
1005        \tex_pdfrefxform:D
1006        \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1007        \scan_stop:
1008    }
1009
1010    \cs_new:Npn \__pdf_backend_xform_ref:n #1
1011    {
1012        \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
1013    }
1014    </pdftex>
1015    <*luatex>
1016    %luatex
1017    %nearly identical but not completely ...
1018    \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1019    % #1 name

```

```

1020 % #2 attributes
1021 % #3 resources
1022 % #4 content, not necessarily a box!
1023 {
1024   \hbox_set:Nn \l__pdf_backend_tmpa_box
1025   {
1026     \bool_set_true:N \l__pdf_backend_xform_bool
1027     \prop_gclear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1028     #4
1029   }
1030   \tl_const:ce
1031   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1032   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1033   \tl_const:ce
1034   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1035   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1036   \tl_const:ce
1037   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1038   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1039   %% do we need to test if #2 and #3 are empty??
1040   \tex_immediate:D \tex_pdfxform:D
1041   ~ attr ~ { #2 }
1042   %% which resources should be default? Is an argument actually needed?
1043   ~ resources ~
1044   {
1045     #3
1046     \int_compare:nNnT
1047       { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1048       >
1049       { 0 }
1050       {
1051         /Properties~
1052         <<
1053         \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1054         >>
1055       }
1056     \prop_if_empty:cF
1057     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1058     {
1059       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1060     }
1061     \prop_if_empty:cF
1062     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1063     {
1064       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1065     }
1066     \prop_if_empty:cF
1067     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1068     {
1069       /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1070     }
1071     \prop_if_empty:cF
1072     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1073     {

```



```

1074         /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1075     }
1076 }
1077 \l__pdf_backend_tmpa_box
1078 \int_const:cn
1079 { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1080 { \tex_pdflastxform:D }
1081 }
1082
1083 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 %protected as with xelatex
1084 {
1085     \tex_pdfrefxform:D \int_use:c
1086     {
1087         c__pdf_backend_xform_ \tl_to_str:n {#1} _int
1088     }
1089     \scan_stop:
1090 }
1091
1092 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1093 { \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
1094
1095 </luatex>
1096 <*dviPDFmx | xdvipdfmx>
1097 % xetex
1098 % it needs a bit testing if it really works to set the box to 0 before the special ...
1099 % does it disturb viewing the xobject?
1100 % what happens with the resources (bdc)? (should work as they are specials too)
1101 % xetex requires that the special is in horizontal mode. This means it affects
1102 % typesetting. But we can no delay the whole form code to shipout
1103 % as the object reference and the size is often wanted on the current page.
1104 % so we need to allocate a box - but probably they won't be thousands xform
1105 % in a document so it shouldn't matter.
1106 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1107 % #1 name
1108 % #2 attributes
1109 % #3 resources
1110 % #4 content, not necessarily a box!
1111 {
1112     \int_gincr:N \g__pdf_backend_object_int
1113     \int_const:cn
1114     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1115     { \g__pdf_backend_object_int }
1116     \box_new:c { g__pdf_backend_xform_#1_box }
1117     \hbox_gset:cn { g__pdf_backend_xform_#1_box }
1118     {
1119         \bool_set_true:N \l__pdf_backend_xform_bool
1120         #4
1121     }
1122     \tl_const:ce
1123     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1124     { \tex_the:D \box_wd:c { g__pdf_backend_xform_#1_box } }
1125     \tl_const:ce
1126     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1127     { \tex_the:D \box_ht:c { g__pdf_backend_xform_#1_box } }

```

```

1128 \tl_const:ce
1129 { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1130 { \tex_the:D \box_dp:c { g__pdf_backend_xform_#1_box } }
1131 \box_set_dp:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1132 \box_set_ht:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1133 \box_set_wd:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1134 \hook_gput_next_code:nn {shipout/background}
1135 {
1136   \mode_leave_vertical: %needed, the xform disappears without it.
1137   \__pdf_backend:e
1138   {
1139     bXObject ~ \__pdf_backend_xform_ref:n { #1 }
1140     \c_space_tl width ~ \pdfxform_wd:n { #1 }
1141     \c_space_tl height ~ \pdfxform_ht:n { #1 }
1142     \c_space_tl depth ~ \pdfxform_dp:n { #1 }
1143   }
1144   \box_use_drop:c { g__pdf_backend_xform_#1_box }
1145   \__pdf_backend:e {put ~ @resources ~<<#3>> }
1146   \__pdf_backend:e
1147   {
1148     put~ @resources ~
1149     <<
1150       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1151     >>
1152   }
1153   \__pdf_backend:e
1154   {
1155     put~ @resources ~
1156     <<
1157       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1158     >>
1159   }
1160   \__pdf_backend:e
1161   {
1162     put~ @resources ~
1163     <<
1164       /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1165     >>
1166   }
1167   \__pdf_backend:e
1168   {
1169     put~ @resources ~
1170     <<
1171       /ColorSpace~
1172       \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1173     >>
1174   }
1175   \__pdf_backend:e {exobj ~<<#2>>}
1176 }
1177 }
1178
1179
1180
1181 \cs_new:Npn \__pdf_backend_xform_ref:n #1

```

```

1182     {
1183       @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1184     }
1185
1186   \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1187   {
1188     \hbox_set:Nn \l__pdf_backend_tmpa_box
1189     {
1190       \__pdf_backend:e
1191       {
1192         uxobj~ \__pdf_backend_xform_ref:n { #1 }
1193       }
1194     }
1195     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1196     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1197     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1198     \box_use_drop:N \l__pdf_backend_tmpa_box
1199   }
1200   </dvipdfmx|xdvipdfmx>
1201   <*dvisvgm>
1202   % unclear what it should do!!
1203   \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}
1204   \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
1205   \cs_new:Npn \__pdf_backend_xform_ref:n {}
1206   </dvisvgm>

```

The xform code for dvips is based on code from the attachfile2 package (in atfi-dvips), along with some ideas from pdfbase and has been corrected with the help of Alexander Grahn. Details like clipping and landscape will probably be corrected in the future. We need some temporary variables to store dimensions

```

1207   <*dvips>
1208   \tl_new:N \l__pdf_backend_xform_tmpwd_tl
1209   \tl_new:N \l__pdf_backend_xform_tmpdp_tl
1210   \tl_new:N \l__pdf_backend_xform_tmphl_tl
1211   \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 % #1 name, #2 attribute, #4
1212   {
1213     \int_gincr:N \g__pdf_backend_object_int
1214     \int_const:cn
1215     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1216     { \g__pdf_backend_object_int }
1217
1218     \hbox_set:Nn \l__pdf_backend_tmpa_box
1219     {
1220       \bool_set_true:N \l__pdf_backend_xform_bool
1221       \prop_gc_clear:c {\__kernel_pdffdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
1222       #4
1223     }
1224     %store the dimensions
1225     \tl_const:ce
1226     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1227     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1228     \tl_const:ce
1229     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1230     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }

```

```

1231 \tl_const:ce
1232 { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1233 { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1234 %store content dimensions in DPI units (Dots) (code from issue 25)
1235 \tl_set:N\l__pdf_backend_xform_tmpwd_tl
1236 {
1237   \dim_to_decimal_in_sp:n{ \box_wd:N \l__pdf_backend_tmpa_box }~
1238   65536~div~72.27~div~DVImag~mul~Resolution~mul~
1239 }
1240 \tl_set:N\l__pdf_backend_xform_tmph_t_tl
1241 {
1242   \dim_to_decimal_in_sp:n{ \box_ht:N \l__pdf_backend_tmpa_box }~
1243   65536~div~72.27~div~DVImag~mul~VResolution~mul~
1244 }
1245 \tl_set:N\l__pdf_backend_xform_tmppd_tl
1246 {
1247   \dim_to_decimal_in_sp:n{ \box_dp:N \l__pdf_backend_tmpa_box }~
1248   65536~div~72.27~div~DVImag~mul~VResolution~mul~
1249 }
1250 % mirror the box
1251 %\box_scale:Nnn \l__pdf_backend_tmpa_box {1} {-1}
1252 \hbox_set:Nn\l__pdf_backend_tmppb_box
1253 {
1254   \__kernel_backend_postscript:e
1255   {
1256     gsave~currentpoint~
1257     initclip~ % restore default clipping path (page device/whole page)
1258     clippath~pathbbox~newpath~pop~pop~
1259     \tl_use:N\l__pdf_backend_xform_tmppd_tl~add~translate~
1260     mark~
1261     /objdef~{ pdf.obj \int_use:N\g__pdf_backend_object_int }\c_space_tl~
1262     /BBox[
1263       0~
1264       \tl_use:N\l__pdf_backend_xform_tmph_t_tl~
1265       \tl_use:N\l__pdf_backend_xform_tmpwd_tl~
1266       \tl_use:N\l__pdf_backend_xform_tmppd_tl~
1267       neg
1268     ]
1269     \str_if_eq:eeF{#1}{}
1270     {
1271       product~(Distiller)~search~{pop~pop~pop~#2}{pop}ifelse~
1272     }
1273     /BP~pdfmark~1~-1~scale~neg~exch~neg~exch~translate
1274   }
1275   \box_use_drop:N\l__pdf_backend_tmpa_box
1276   \__kernel_backend_postscript:n
1277   {
1278     mark ~ /EP~pdfmark ~ grestore
1279   }
1280   \str_if_eq:eeF{#1}{}
1281   {
1282     \__kernel_backend_postscript:e
1283     {
1284       product~(Ghostscript)~search~

```

```

1285         {
1286             pop~pop~pop~
1287             mark~
1288             { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1289             ~<<#2>>~/PUT~pdfmark
1290         }{pop}ifelse
1291     }
1292 }
1293 }
1294 \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1295 \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1296 \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1297 \hook_gput_code:nnn {begindocument/end}{pdfxform}
1298 {
1299     \mode_leave_vertical:
1300     \box_use:N\l__pdf_backend_tmpb_box
1301 }
1302 }
1303
1304
1305 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1306 {
1307     \hbox_set:Nn \l__pdf_backend_tmpa_box
1308     {
1309         \__kernel_backend_postscript:e
1310         {
1311             gsave~currentpoint~translate~1~-1~scale~
1312             mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }~
1313             /SP~pdfmark ~ grestore
1314         }
1315     }
1316     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1317     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1318     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1319     \box_use_drop:N \l__pdf_backend_tmpa_box
1320 }
1321 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1322 {
1323     { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1324 }
1325
1326 </dvips>
1327 <*drivers>
1328 %% all
1329 \prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
1330 {
1331     \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1332     { \prg_return_true: }
1333     { \prg_return_false: }
1334 }
1335 \prg_new_eq_conditional:NNn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
1336 { TF , T , F , p }
1337 </drivers>

```

(End of definition for __pdf_backend_xform_new:nnnn, __pdf_backend_xform_use:n, and __pdf_

`backend_xform_ref:n.)`

1.11 Structure Destinations

Standard destinations consist of a reference to a page in the pdf and instructions how to display it—typically they will put a specific location in the left top corner of the viewer and so give the impression that a link jumped to the word in this place. But in reality they are not connected to the content.

Starting with pdf 2.0 destinations can in a tagged PDF also point to a structure, to a `/StructElem` object. `GoTo` links can then additionally to the `/D` key pointing to a page destination also point to such a structure destination with an `/SD` key. Programs that e.g. convert such a PDF to html can then create better links. (According to the reference, PDF-viewer should prefer the structure destination over the page destination, but as far as it is known this isn't done yet.)

Currently structure destinations and `GoTo` links making use of it could natively only be created with the `dvipdfmx` backend. With `pdftex` and `lualatex` it was only possible to create a restricted type which used only the “Fit” mode. Starting with `TeXlive 2022` (earlier in `miktex`) both engine will know new keywords which allow to create structure destination easily.

The following backend code prepares the use of structure destinations. The general idea is that if structure destinations are used, they should be used always. So we define alternative commands which can be activated by mapping them to the standard backend commands.

The needed code differ depending on if structure objects use standard or indexed object names. At the end we will probably always use indexed objects, but for now we offer both options.

`\l_pdf_current_structure_destination_tl`

This command holds the name of the structure object to use in the following commands which creates a destination. The code which activates structure destinations must also ensure that it has a sensible, expandable content. `tagpdf` for example will define it as

```
\tl_set:Nn \l_pdf_current_structure_destination_tl { __tag/struct/\g__tag_struct_stack
or if indexed structure object names are used
```

```
\tl_set:Nn \l_pdf_current_structure_destination_tl { {__tag/struct}{\g__tag_struct_sta
```

```
1338 <*drivers>
```

```
1339 \tl_new:N \l_pdf_current_structure_destination_tl
```

```
1340 </drivers>
```

(End of definition for `\l_pdf_current_structure_destination_tl`.)

We will define alternatives for three backend commands:

```
\__pdf_backend_destination:nn      -> \__pdf_backend_structure_destination:nn
\__pdf_backend_destination:nnnn -> \__pdf_backend_structure_destination:nnnn
\__pdfannot_backend_link_begin_goto:nnw -> \__pdf_backend_link_begin_structure_goto:nnw
\__pdf_backend_destination:nn      -> \__pdf_backend_indexed_structure_destination:nn
\__pdf_backend_destination:nnnn -> \__pdf_backend_indexed_structure_destination:nnnn
\__pdfannot_backend_link_begin_goto:nnw -> \__pdf_backend_indexed_link_begin_structur
```

Activating means mapping them onto the original commands. Be aware that not all engines and compilation routes support structure destinations, for them the command will be a no-op.

```

\pdf_activate_structure_destination:
pdf_activate_indexed_structure_destination:
1341 <*drivers>
1342 \cs_new_protected:Npn \pdf_activate_structure_destination:
1343 {
1344   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_structure_destination:nn
1345   \cs_gset_eq:NN \__pdf_backend_destination:nnnn \__pdf_backend_structure_destination:nnnn
1346   \cs_gset_eq:NN \__pdfannot_backend_link_begin_goto:nnw \__pdfannot_backend_link_begin_goto:nnw
1347 }
1348 \cs_new_protected:Npn \pdf_activate_indexed_structure_destination:
1349 {
1350   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_indexed_structure_destination:nn
1351   \cs_gset_eq:NN \__pdf_backend_destination:nnnn \__pdf_backend_indexed_structure_destination:nnnn
1352   \cs_gset_eq:NN \__pdfannot_backend_link_begin_goto:nnw \__pdfannot_backend_link_begin_goto:nnw
1353 }
1354 </drivers>

```

(End of definition for \pdf_activate_structure_destination: and \pdf_activate_indexed_structure_destination:.)

Now the driver dependent parts. By default the new commands are simply copies of the original commands. We adapt them then for the engines and engine version which provide support for structure destinations.

```

1355 <*drivers>
1356 \cs_set_eq:NN \__pdf_backend_structure_destination:nn \__pdf_backend_destination:nn
1357 \cs_set_eq:NN \__pdf_backend_structure_destination:nnnn \__pdf_backend_destination:nnnn
1358 \cs_set_eq:NN \__pdfannot_backend_link_begin_structure_goto:nnw \__pdfannot_backend_link_begin_goto:nnw
1359 \cs_set_eq:NN \__pdf_backend_indexed_structure_destination:nn \__pdf_backend_destination:nn
1360 \cs_set_eq:NN \__pdf_backend_indexed_structure_destination:nnnn \__pdf_backend_destination:nnnn
1361 </drivers>

```

```

\__pdf_backend_structure_destination:nn
\__pdf_backend_structure_destination:nnnn
\__pdfannot_backend_link_begin_structure_goto:nnw

```

These commands are the backend commands to create a destination. which create also a structure destination. At first xetex/dvipdfmx. The structure destination is an array, so we use obj for it so that we can reference it:

```

1362 <*xdvipdfmx|dvipdfmx>
1363 \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1364 {
1365   \__pdf_backend:e
1366   {
1367     dest ~ ( \exp_not:n {#1} )
1368     [
1369       @thispage
1370       \str_case:nnF {#2}
1371       {
1372         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1373         { fit } { /Fit }
1374         { fitb } { /FitB }
1375         { fitbh } { /FitBH }
1376         { fitbv } { /FitBV ~ @xpos }
1377         { fith } { /FitH ~ @ypos }
1378         { fitv } { /FitV ~ @xpos }
1379         { fitr } { /Fit }
1380       }
1381       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1382     ]
1383   }

```

We test if the structure object exist. The object of the structure destination gets the name `@pdf.Sdest.<destname>`, where `<destname>` is the name of the standard destination so that we can reference it in the GoTo links.

```

1384 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1385 {
1386   \__pdf_backend:e
1387   {
1388     obj ~ @pdf.SDest.\exp_not:n{#1}
1389     [
1390       \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1391       \str_case:nnF {#2}
1392       {
1393         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1394         { fit } { /Fit }
1395         { fitb } { /FitB }
1396         { fitbh } { /FitBH }
1397         { fitbv } { /FitBV ~ @xpos }
1398         { fith } { /FitH ~ @ypos }
1399         { fitv } { /FitV ~ @xpos }
1400         { fitr } { /Fit }
1401       }
1402       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1403     ]
1404   }
1405 }
1406 }

```

The second destination command is for the boxed destination. Here we need to define an new auxiliary command:

```

1407 \cs_new_protected:Npn \__pdf_backend_structure_destination_aux:nnnn #1#2#3#4
1408 {
1409   \vbox_to_zero:n
1410   {
1411     \__kernel_kern:n {#4}
1412     \hbox:n
1413     {
1414       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1415       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1416     }
1417     \tex_vss:D
1418   }
1419   \__kernel_kern:n {#1}
1420   \vbox_to_zero:n
1421   {
1422     \__kernel_kern:n { -#3 }
1423     \hbox:n
1424     {
1425       \__pdf_backend:n
1426       {
1427         dest ~ (#2)
1428         [
1429           @thispage
1430           /FitR ~
1431           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~

```



```

1432         @xpos ~ @ypos
1433     ]
1434 }

```

Here we add the structure destination to the same box

```

1435     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1436     {
1437         \__pdf_backend:e
1438         {
1439             obj ~ @pdf.SDest.\exp_not:n{#2}
1440             [
1441                 \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_
1442                 /FitR ~
1443                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1444                 @xpos ~ @ypos
1445             ]
1446         }
1447     }
1448 }
1449 \tex_vss:D
1450 }
1451 \__kernel_kern:n { -#1 }
1452 }

```

And now we redefine the destination command:

```

1453 \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1454 {
1455     \exp_args:Ne \__pdf_backend_structure_destination_aux:nnnn
1456     { \dim_eval:n {#2} } {#1} {#3} {#4}
1457 }

```

At last the goto link.

```

1458 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1459 {
1460     \__pdfannot_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD-@pdf.S
1461     }
1462     </xdvipdfmx|dvipdfmx>

```

Now pdftex. We only redefine for version 1.40 revision 24 or later.

```

1463 <*pdftex>
1464 \bool_lazy_and:nnT
1465 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1466 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
1467 {
1468     \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1469     {
1470         \tex_pdfdest:D
1471         name {#1}
1472         \str_case:nnF {#2}
1473         {
1474             { xyz } { xyz }
1475             { fit } { fit }
1476             { fitb } { fitb }
1477             { fitbh } { fitbh }
1478             { fitbv } { fitbv }
1479             { fith } { fith }

```

```

1480         { fitv } { fitv }
1481         { fitr } { fitr }
1482     }
1483     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1484     \scan_stop:
1485     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1486     {
1487         \tex_pdfdest:D
1488         struct~
1489         \int_use:c
1490         { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl}
1491         name {#1}
1492         \str_case:nnF {#2}
1493         {
1494             { xyz } { xyz }
1495             { fit } { fit }
1496             { fitb } { fitb }
1497             { fitbh } { fitbh }
1498             { fitbv } { fitbv }
1499             { fith } { fith }
1500             { fitv } { fitv }
1501             { fitr } { fitr }
1502         }
1503         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1504         \scan_stop:
1505     }
1506 }
1507 \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1508 {
1509     \tex_pdfdest:D
1510     name {#1}
1511     fitr ~
1512     width \dim_eval:n {#2} ~
1513     height \dim_eval:n {#3} ~
1514     depth \dim_eval:n {#4} \scan_stop:
1515     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1516     {
1517         \tex_pdfdest:D
1518         struct~
1519         \int_use:c
1520         { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl}
1521         name {#1}
1522         fitr ~
1523         width \dim_eval:n {#2} ~
1524         height \dim_eval:n {#3} ~
1525         depth \dim_eval:n {#4} \scan_stop:
1526     }
1527 }
1528 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1529 {
1530     \__pdfannot_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1531 }
1532 }
1533 </pdftex>

```

luatex is quite similar to pdftex. Mostly the test for the version is different

```

1534 (*luatex)
1535 \int_compare:nNtT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1536 {
1537   \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1538   {
1539     \tex_pdfextension:D dest
1540     name {#1}
1541     \str_case:nnF {#2}
1542     {
1543       { xyz } { xyz }
1544       { fit } { fit }
1545       { fitb } { fitb }
1546       { fitbh } { fitbh }
1547       { fitbv } { fitbv }
1548       { fith } { fith }
1549       { fitv } { fitv }
1550       { fitr } { fitr }
1551     }
1552     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1553     \scan_stop:
1554     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1555     {
1556       \tex_pdfextension:D dest
1557       struct~
1558       \int_use:c
1559       { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl} }
1560       name {#1}
1561       \str_case:nnF {#2}
1562       {
1563         { xyz } { xyz }
1564         { fit } { fit }
1565         { fitb } { fitb }
1566         { fitbh } { fitbh }
1567         { fitbv } { fitbv }
1568         { fith } { fith }
1569         { fitv } { fitv }
1570         { fitr } { fitr }
1571       }
1572       { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1573       \scan_stop:
1574     }
1575   }
1576   \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1577   {
1578     \tex_pdfextension:D dest
1579     name {#1}
1580     fitr ~
1581     width \dim_eval:n {#2} ~
1582     height \dim_eval:n {#3} ~
1583     depth \dim_eval:n {#4} \scan_stop:
1584     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1585     {
1586       \tex_pdfextension:D dest

```

```

1587         struct~
1588         \int_use:c
1589         { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination}
1590         name {#1}
1591         fitr ~
1592         width \dim_eval:n {#2} ~
1593         height \dim_eval:n {#3} ~
1594         depth \dim_eval:n {#4} \scan_stop:
1595     }
1596 }
1597 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1598 {
1599     \__pdfannot_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1600 }
1601 }
1602 </luatex>

```

(End of definition for __pdf_backend_structure_destination:nn, __pdf_backend_structure_destination:nnnn, and __pdfannot_backend_link_begin_structure_goto:nnw.)

This are the indexed variants of the commands to create a destination and a structure destination. At first xetex/dvipdfmx. The structure destination is an array, so we use obj for it so that we can reference it:

```

1603 < *xdvipdfmx | dvipdfmx >
1604 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nn #1#2
1605 {
1606     \__pdf_backend:e
1607     {
1608         dest ~ ( \exp_not:n {#1} )
1609         [
1610             @thispage
1611             \str_case:nnF {#2}
1612             {
1613                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1614                 { fit } { /Fit }
1615                 { fitb } { /FitB }
1616                 { fitbh } { /FitBH }
1617                 { fitbv } { /FitBV ~ @xpos }
1618                 { fith } { /FitH ~ @ypos }
1619                 { fitv } { /FitV ~ @xpos }
1620                 { fitr } { /Fit }
1621             }
1622             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1623         ]
1624     }

```

We do not test anymore if the structure object exist. The object of the structure destination gets the name @pdf.Sdest.<destname>, where <destname> is the name of the standard destination so that we can reference it in the GoTo links.

```

1625     \__pdf_backend:e
1626     {
1627         obj ~ @pdf.SDest.\exp_not:n{#1}
1628         [
1629             \exp_after:wN \pdf_object_ref_indexed:nn \l_pdf_current_structure_destination_t

```

```

1630 \str_case:nnF {#2}
1631 {
1632   { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1633   { fit } { /Fit }
1634   { fitb } { /FitB }
1635   { fitbh } { /FitBH }
1636   { fitbv } { /FitBV ~ @xpos }
1637   { fith } { /FitH ~ @ypos }
1638   { fitv } { /FitV ~ @xpos }
1639   { fitr } { /Fit }
1640 }
1641 { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1642 ]
1643 }
1644 }

```

The second destination command is for the boxed destination. Here we need to define an new auxiliary command:

```

1645 \cs_new_protected:Npn \__pdf_backend_indexed_structure_destination_aux:nnnn #1#2#3#4
1646 {
1647   \vbox_to_zero:n
1648   {
1649     \__kernel_kern:n {#4}
1650     \hbox:n
1651     {
1652       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1653       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1654     }
1655     \tex_vss:D
1656   }
1657   \__kernel_kern:n {#1}
1658   \vbox_to_zero:n
1659   {
1660     \__kernel_kern:n { -#3 }
1661     \hbox:n
1662     {
1663       \__pdf_backend:n
1664       {
1665         dest ~ (#2)
1666         [
1667           @thispage
1668           /FitR ~
1669           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1670           @xpos ~ @ypos
1671         ]
1672       }
1673     }
1674   }

```

Here we add the structure destination to the same box

```

1673 \__pdf_backend:e
1674 {
1675   obj ~ @pdf.SDest.\exp_not:n{#2}
1676   [
1677     \exp_after:wN \pdf_object_ref_indexed:nn \l_pdf_current_structure_destin
1678     /FitR ~
1679     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~

```

```

1680             @xpos ~ @ypos
1681         ]
1682     }
1683 }
1684 \tex_vss:D
1685 }
1686 \__kernel_kern:n { -#1 }
1687 }

```

And now we redefine the destination command:

```

1688 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1689 {
1690     \exp_args:Ne \__pdf_backend_indexed_structure_destination_aux:nnnn
1691     { \dim_eval:n {#2} } {#1} {#3} {#4}
1692 }
1693 </xdvipdfmx|dvipdfmx>

```

Now pdftex. We only redefine for version 1.40 revision 24 or later.

```

1694 <*pdftex>
1695 \bool_lazy_and:nnT
1696 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1697 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
1698 {
1699     \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nn #1#2
1700     {
1701         \tex_pdfdest:D
1702         name {#1}
1703         \str_case:nnF {#2}
1704         {
1705             { xyz } { xyz }
1706             { fit } { fit }
1707             { fitb } { fitb }
1708             { fitbh } { fitbh }
1709             { fitbv } { fitbv }
1710             { fith } { fith }
1711             { fitv } { fitv }
1712             { fitr } { fitr }
1713         }
1714         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1715     }
1716     \scan_stop:
1717     \tex_pdfdest:D
1718     struct~
1719     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_des
1720     name {#1}
1721     \str_case:nnF {#2}
1722     {
1723         { xyz } { xyz }
1724         { fit } { fit }
1725         { fitb } { fitb }
1726         { fitbh } { fitbh }
1727         { fitbv } { fitbv }
1728         { fith } { fith }
1729         { fitv } { fitv }
1730         { fitr } { fitr }
1731     }

```

```

1731         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1732         \scan_stop:
1733     }
1734     \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1735     {
1736         \tex_pdfdest:D
1737         name {#1}
1738         fitr ~
1739         width \dim_eval:n {#2} ~
1740         height \dim_eval:n {#3} ~
1741         depth \dim_eval:n {#4} \scan_stop:
1742     \tex_pdfdest:D
1743     struct~
1744     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_destinati
1745     name {#1}
1746     fitr ~
1747     width \dim_eval:n {#2} ~
1748     height \dim_eval:n {#3} ~
1749     depth \dim_eval:n {#4} \scan_stop:
1750 }
1751 }
1752 </pdftex>

```

luatex is quite similar to pdftex. Mostly the test for the version is different

```

1753 <*luatex>
1754 \int_compare:nNnT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1755 {
1756     \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nn #1#2
1757     {
1758         \tex_pdfextension:D dest
1759         name {#1}
1760         \str_case:nnF {#2}
1761         {
1762             { xyz } { xyz }
1763             { fit } { fit }
1764             { fitb } { fitb }
1765             { fitbh } { fitbh }
1766             { fitbv } { fitbv }
1767             { fith } { fith }
1768             { fitv } { fitv }
1769             { fitr } { fitr }
1770         }
1771         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1772         \scan_stop:
1773     \tex_pdfextension:D dest
1774     struct~
1775     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_desti
1776     name {#1}
1777     \str_case:nnF {#2}
1778     {
1779         { xyz } { xyz }
1780         { fit } { fit }
1781         { fitb } { fitb }
1782         { fitbh } { fitbh }
1783         { fitbv } { fitbv }

```

```

1784         { fith } { fith }
1785         { fitv } { fitv }
1786         { fitr } { fitr }
1787     }
1788     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1789     \scan_stop:
1790 }
1791 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1792 {
1793     \tex_pdfextension:D dest
1794     name {#1}
1795     fitr ~
1796     width \dim_eval:n {#2} ~
1797     height \dim_eval:n {#3} ~
1798     depth \dim_eval:n {#4} \scan_stop:
1799     \tex_pdfextension:D dest
1800     struct~
1801     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_destinati
1802     name {#1}
1803     fitr ~
1804     width \dim_eval:n {#2} ~
1805     height \dim_eval:n {#3} ~
1806     depth \dim_eval:n {#4} \scan_stop:
1807 }
1808 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1809 {
1810     \__pdfannot_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1811 }
1812 }
1813 </luatex>

```

(End of definition for __pdf_backend_indexed_structure_destination:nn and __pdf_backend_indexed_structure_destination:nnnn.)

1.12 Settings for regression tests

When doing pdf based regression tests some meta data in the pdf should have fixed values to get identical pdf's. We define here the backend dependent part. The main command is then in l3pdfmeta

```

1814 <*drivers>
1815 \cs_new_protected:Npn \__pdf_backend_set_regression_data:
1816 {
1817     \sys_gset_rand_seed:n{1000}
1818     \pdfmanagement_add:nnn{Info}{Creator}{(TeX)}
1819 </drivers>
1820 <*dvips>
1821     \AddToHook{begindocument}{\pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX+dvips)}}
1822     \__kernel_backend_literal:e{!~<</DocumentUUID~(DocumentUUID)>>~setpagedevice}
1823     \__kernel_backend_literal:e{!~<</InstanceUUID~(InstanceUUID)>>~setpagedevice}
1824     \pdfmanagement_add:nne{Info}{CreationDate}{(\c_sys_timestamp_str)}
1825     \pdfmanagement_add:nne{Info}{ModDate}{(\c_sys_timestamp_str)}
1826 </dvips>
1827 <*dviPDFmx>
1828     \pdfmanagement_add:nnn{Info}{Producer}{(dviPDFmx)}

```



```

1829 \__kernel_backend_literal:e
1830 {pdf:trailerid [~
1831 <00112233445566778899aabbccddeeff>~
1832 <00112233445566778899aabbccddeeff>~
1833 ]}
1834 </dviPDFmx>
1835 <*xdiPDFmx>
1836 \pdfmanagement_add:nnn{Info}{Producer}{(xetex)}
1837 \__kernel_backend_literal:e
1838 {pdf:trailerid [~
1839 <00112233445566778899aabbccddeeff>~
1840 <00112233445566778899aabbccddeeff>~
1841 ]}
1842 </xdiPDFmx>
1843 <*pdftex>
1844 \pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX)}
1845 \tex_pdfsuppressptexinfo:D 7 \scan_stop:
1846 \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
1847 </pdftex>
1848 <*luatex>
1849 \pdfmanagement_add:nnn{Info}{Producer}{(LuaTeX)}
1850 \tex_pdfvariable:D suppressoptionalinfo 7\relax
1851 \tex_pdfvariable:D trailerid
1852 {[~
1853 <2350CAD05F8A7AF0AA4058486855344F>~
1854 <2350CAD05F8A7AF0AA4058486855344F>~
1855 ]}
1856 </luatex>

```

Embedded files should also have a fix date.

```

1857 <*drivers>
1858 \pdfdict_put:nne {l_pdffile/Params} {ModDate}{(\c_sys_timestamp_str)}
1859 \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-
8c19-6237d832fc80}
1860 \AddToDocumentProperties[hyperref]{pdfproducer}{\c_sys_engine_exec_str-NN.NN.NN}
1861 }
1862 </drivers>

```

1.13 Uncompressed metadata object stream

The xmp metadata should be written “uncompressed” to pdf. It is not quite clear what exactly that means. Probably it only means that there should be no `/Filter` key in the stream, but packages like `pdfx` and `hyperref` try to suppress object compression too, so we add support for it too. With `luatex` this is possible by using the `uncompressed` key word. With `pdftex` one can change locally the `compresslevel`. `(x)diPDFmx` does it automatically and doesn’t need some special command. No solution is known for the `dvips` route. We need it only once, so we make it special and probably no public interface is needed. It writes the `__pdfmeta/xmp` object which should be declared before.

`luatex` has of now (2025-11-12) a bug: using the `uncompressed` key disables object compression for all following objects. We therefor delay the writing into the `enddocument/end` hook after the `tagpdf` code.

```

1863 <*luatex>
1864 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1

```

```

1865 {
1866   \AddToHook{enddocument/end}
1867   {
1868     \tex_immediate:D \tex_pdfextension:D obj ~
1869     useobjnum ~ \int_eval:n{\__pdf_object_retrieve:n {\__pdfmeta/xmp}}~uncompressed~
1870     \__pdf_backend_object_write:nn {stream}
1871     {{/Type~/Metadata~/Subtype~/XML}{#1}}
1872   }
1873 }
1874 </luatex>
1875 <*pdftex>
1876 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1877 {
1878   \group_begin:
1879   \tex_pdfcompresslevel:D 0 \scan_stop:
1880   \tex_immediate:D \tex_pdfobj:D useobjnum ~ \int_eval:n{\__pdf_object_retrieve:n
1881   {\__pdfmeta/xmp}}~
1882   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1883   \group_end:
1884 }
1885 </pdftex>
1886 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1887 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1888 {
1889   \pdf_object_write:nnn{\__pdfmeta/xmp} {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
1890 }
1891 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>

```

1.14 Suppressing deprecated PDF features

/ProcSet, /CharSet and the /Info dictionary are deprecated in PDF 2.0. For the pdf/A-4 standard they must be suppressed. Not every engine is able to do this, but for pdfTeX and luatex we define suitable backend command. /ProcSet is suppressed automatically for pdf version 2.0 starting with in texlive 2023.

`__pdf_backend_omit_charset:n` The option to omit /Charset exists already for quite some time for the two engines.

```

1892 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1893 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 {} % #1 number
1894 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1895 <*pdftex>
1896 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 % #1 number
1897 {
1898   \tex_pdfomitcharset:D = #1 \scan_stop:
1899 }
1900 </pdftex>
1901 <*luatex>
1902 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 % #1 number
1903 {
1904   \tex_pdfvariable:D omitcharset = #1 \scan_stop:
1905 }
1906 </luatex>

```

(End of definition for __pdf_backend_omit_charset:n.)

`_pdf_backend_omit_info:n` The option to suppress the info dictionary will be available in texlive 2023.

```

1907 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1908 \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 {} % #1 number
1909 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1910 <*pdfTeX>
1911 \bool_lazy_and:nnTF
1912 { \int_compare_p:nNn {\tex_pdfTeXversion:D } > {139} }
1913 { \int_compare_p:nNn {\tex_pdfTeXrevision:D } > {24} }
1914 {
1915   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 % #1 number
1916   {
1917     \pdfomitinfodict = #1 \scan_stop:
1918   }
1919 }
1920 {
1921   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 {} % #1 number
1922 }
1923 }
1924 </pdfTeX>
1925 <*luatex>
1926 \int_compare:nNnTF {\directlua{tex.print(status.list()["development_id"])} } > {7560}
1927 {
1928   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 % #1 number
1929   {
1930     \tex_pdfvariable:D omitinfodict = #1 \scan_stop:
1931   }
1932 }
1933 {
1934   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 {} % #1 number
1935 }
1936 </luatex>

```

(End of definition for _pdf_backend_omit_info:n.)

With luatex it is for some standards also necessary to suppress the CidSet entry in the fonts (with xetex there seem to be no problem).

`_pdf_backend_omit_cidset:n` The option to omit /CharSet exists already for quite some time for the two engines.

```

1937 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm | pdfTeX>
1938 \cs_new_protected:Npn \_pdf_backend_omit_cidset:n #1 {} % #1 number
1939 </xdvipdfmx | dvipdfmx | dvips | dvisvgm | pdfTeX>
1940 <*luatex>
1941 \cs_new_protected:Npn \_pdf_backend_omit_cidset:n #1 % #1 number
1942 {
1943   \tex_pdfvariable:D omitcidset = #1 \scan_stop:
1944 }
1945 </luatex>

```

(End of definition for _pdf_backend_omit_cidset:n.)

1.15 Outline backend

```

1946 <*dvips>
1947 \cs_new_protected:Npn \_pdf_backend_action:nnnnnn #1#2#3#4#5#6#7
1948 % #1 level (ignored), #2 kid count, #3 open? #4 color (space separated values), #5 flag, #6

```

```

1949 {
1950   \__pdf_backend_pdfmark:e
1951   {
1952     /Title~(\exp_not:n{#7})~
1953     /Count~\bool_if:nF{#3}{-}#2~
1954     \tl_if_blank:nF { #4 }{/C~[#4]}
1955     /F~#5
1956     /Action <<\exp_not:n{#6}>>
1957     /OUT
1958   }
1959 }
1960 \cs_new_protected:Npn \__pdf_backend_goto:nnnnnnn #1#2#3#4#5#6#7
1961 % #1 level (ignored), #2 kid count, #3 open? #4 color (space separated values), #5 flag, #6
1962 {
1963   \__pdf_backend_pdfmark:e
1964   {
1965     /Title~(\exp_not:n{#7})~
1966     /Count~\bool_if:nF{#3}{-}#2~
1967     \tl_if_blank:nF { #4 }{/C~[#4]}
1968     /F~#5
1969     /Action/GoTo/Dest(\exp_not:n{#6})~cvm
1970     /OUT
1971   }
1972 }
1973 </dvips>
1974 <*pdftex>
1975 \cs_new_protected:Npn \__pdf_backend_action:nnnnnnn #1#2#3#4#5#6#7
1976 % #1 level (ignored), #2 kid count, #3 open? #4 color (space separated values), #5 flag, #6
1977 {
1978   \tex_pdfoutline:D~attr
1979   {
1980     /F~#5~
1981     \tl_if_blank:nF { #4 }{/C~[#4]}
1982   }~
1983   user~{<<\exp_not:n{#6}>>}~
1984   count~\bool_if:nF{#3}{-}#2~
1985   {\exp_not:n{#7}}
1986 }
1987 \cs_new_protected:Npn \__pdf_backend_goto:nnnnnnn #1#2#3#4#5#6#7
1988 % #1 level (ignored), #2 kid count, #3 open? #4 color (space separated values), #5 flag, #6
1989 {
1990   \tex_pdfoutline:D~attr
1991   {
1992     /F~#5~
1993     \tl_if_blank:nF { #4 }{/C~[#4]}
1994   }~
1995   \tag_if_active:TF
1996   {goto~struct~name~{#6}~name~{#6}}
1997   {goto~name~{#6}}~
1998   count~\bool_if:nF{#3}{-}#2~
1999   {\exp_not:n{#7}}
2000 }
2001 </pdftex>
2002 <*luatex>

```

```

2003 \cs_new_protected:Npn \__pdf_backend_action:nnnnnnn #1#2#3#4#5#6#7
2004 % #1 level (ignored), #2 kid count, #3 open? #4 color (space separated values), #5 flag, #6
2005 {
2006   \tex_pdfextension:D~outline~attr
2007   {
2008     /F~#5~
2009     \tl_if_blank:nF { #4 }{/C~[#4]}
2010   }~
2011   user~{<<\exp_not:n{#6}>>}~
2012   count~\bool_if:nF{#3}{-}#2~
2013   {\exp_not:n{#7}}
2014 }
2015 \cs_new_protected:Npn \__pdf_backend_goto:nnnnnnn #1#2#3#4#5#6#7
2016 % #1 level (ignored), #2 kid count, #3 open? #4 color (space separated values), #5 flag, #6
2017 {
2018   \tex_pdfextension:D~outline~attr
2019   {
2020     /F~#5~
2021     \tl_if_blank:nF { #4 }{/Color~[#4]}
2022   }~
2023   \tag_if_active:TF
2024   {goto~struct~name~{#6}~name~{#6}}
2025   {goto~name~{#6}}~
2026   count~\bool_if:nF{#3}{-}#2~
2027   {\exp_not:n{#7}}
2028 }
2029 </luatex>
2030 <*xdvipdfmx|dvipdfmx>
2031 \cs_new_protected:Npn \__pdf_backend_action:nnnnnnn #1#2#3#4#5#6#7
2032 % #1 level, #2 kid count (ignored), #3 open? #4 color (space separated values), #5 flag, #6
2033 {
2034   \tex_special:D
2035   {
2036     pdf:out~
2037     [\bool_if:nF{#3}{-}]~#1~
2038     <<
2039     /A<<\exp_not:n{#6}>>~
2040     /Title~(\exp_not:n{#7})~
2041     /F~#5~
2042     \tl_if_blank:nF { #4 }{/C~[#4]}
2043     >>
2044   }
2045 }
2046 \cs_new_protected:Npn \__pdf_backend_goto:nnnnnnn #1#2#3#4#5#6#7
2047 % #1 level, #2 kid count (ignored), #3 open? #4 color (space separated values), #5 flag, #6
2048 {
2049   \tex_special:D
2050   {
2051     pdf:out~
2052     [\bool_if:nF{#3}{-}]~#1~
2053     <<
2054     /A
2055     <<
2056     /S/GoTo/D(\exp_not:n{#6})

```

```

2057         \tag_if_active:T
2058         {
2059             /SD~@pdf.SDest.#6
2060         }
2061         >>~
2062         /Title~(\exp_not:n{#7})~
2063         /F~#5~
2064         \tl_if_blank:nF { #4 }{/C~[#4]}
2065     >>
2066 }
2067 }
2068 </xdvipdfmx | dvipdfmx>
2069 <*dvisvgm>
2070 \cs_new_protected:Npn \__pdf_backend_action:nnnnnnn #1#2#3#4#5#6#7{}
2071 \cs_new_protected:Npn \__pdf_backend_goto:nnnnnnn #1#2#3#4#5#6#7 {}
2072 </dvisvgm>

```

1.16 lua code for lualatex

```

2073 <lua>
2074 ltx= ltx or {}
2075 ltx.__pdf      = ltx.__pdf or {}
2076 ltx.__pdf.Page = ltx.__pdf.Page or {}
2077 ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
2078 ltx.__pdf.Page.Resources = ltx.__pdf.Page.Resources or {}
2079 ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
2080 ltx.__pdf.Page.Resources.List={"ExtGState","ColorSpace","Pattern","Shading"}
2081 ltx.__pdf.object = ltx.__pdf.object or {}
2082
2083 ltx.pdf= ltx.pdf or {} -- for "public" functions
2084
2085 local __pdf = ltx.__pdf
2086 local pdf = pdf
2087
2088 local function __pdf_backend_Page_gput (name,value)
2089     __pdf.Page.dflt[name]=value
2090 end
2091
2092 local function __pdf_backend_Page_gremove (name)
2093     __pdf.Page.dflt[name]=nil
2094 end
2095
2096 local function __pdf_backend_Page_gclear ()
2097     __pdf.Page.dflt={}
2098 end
2099
2100 local function __pdf_backend_ThisPage_gput (page,name,value)
2101     __pdf.Page[page] = __pdf.Page[page] or {}
2102     __pdf.Page[page][name]=value
2103 end
2104
2105 local function __pdf_backend_ThisPage_gpush (page)
2106     local token=""
2107     local t = {}
2108     local tkeys= {}

```

```

2109 for name,value in pairs(__pdf.Page.dflt) do
2110     t[name]=value
2111 end
2112 if __pdf.Page[page] then
2113     for name,value in pairs(__pdf.Page[page]) do
2114         t[name] = value
2115     end
2116 end
2117 -- sort the table to get reliable test files.
2118 for name,value in pairs(t) do
2119     table.insert(tkeys,name)
2120 end
2121 table.sort(tkeys)
2122 for _,name in ipairs(tkeys) do
2123     token = token .. "/"..name.." "..t[name]
2124 end
2125 return token
2126 end
2127
2128 function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly_
2129 __pdf_backend_ThisPage_gput (page,name,value)
2130 end
2131
2132 function ltx.__pdf.backend_ThisPage_gpush (page)
2133     pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
2134 end
2135
2136 function ltx.__pdf.backend_Page_gput (name,value)
2137     __pdf_backend_Page_gput (name,value)
2138 end
2139
2140 function ltx.__pdf.backend_Page_gremove (name)
2141     __pdf_backend_Page_gremove (name)
2142 end
2143
2144 function ltx.__pdf.backend_Page_gclear ()
2145     __pdf_backend_Page_gclear ()
2146 end
2147
2148
2149 local Properties = ltx.__pdf.Page.Resources.Properties
2150 local ResourceList= ltx.__pdf.Page.Resources.List
2151 local function __pdf_backend_PageResources_gpush (page)
2152     local token=""
2153     if Properties[page] then
2154         -- we sort the table, so that the pdf test works
2155         local t = {}
2156         for name,value in pairs (Properties[page]) do
2157             table.insert (t,name)
2158         end
2159         table.sort (t)
2160         for _,name in ipairs(t) do
2161             token = token .. "/"..name.." ".. Properties[page][name]
2162         end

```

```

2163 token = "/Properties <<\"..token..\">>"
2164 end
2165 for i,name in ipairs(ResourceList) do
2166   if ltx.__pdf.Page.Resources[name] then
2167     token = token .. "/" .. name .. " " .. ltx.pdf.object_ref("__pdf/Page/Resources/" .. name)
2168   end
2169 end
2170 return token
2171 end
2172
2173 -- the function is public, as I probably need it in tagpdf too ...
2174 function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
2175   Properties[page] = Properties[page] or {}
2176   Properties[page][name]=value
2177   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
2178 end
2179
2180 function ltx.pdf.Page_Resources_gpush(page)
2181   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
2182 end
2183
2184 function ltx.pdf.object_ref (objname)
2185   if ltx.__pdf.object[objname] then
2186     local ref= ltx.__pdf.object[objname]
2187     return ref
2188   else
2189     return "false"
2190   end
2191 end
2192 </lua>

```

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>\AddToDocumentProperties</code>	1859, 1860
<code>\AddToHook</code>	1821, 1866
<code>\AssignSocketPlug</code>	933
B	
bool commands:	
<code>\bool_if:N</code> TF	672, 696, 752, 782
<code>\bool_if:n</code> TF	1953, 1966,
1984, 1998, 2012, 2026, 2037, 2052	
<code>\bool_lazy_and:nn</code> TF .	1464, 1695, 1911
<code>\bool_new:N</code>	512
<code>\bool_set_true:N</code> .	944, 1026, 1119, 1220
box commands:	
<code>\box_dp:N</code> .	957, 1038, 1130, 1233, 1247
<code>\box_ht:N</code> .	954, 1035, 1127, 1230, 1242
<code>\box_new:N</code>	85, 86, 1116
<code>\box_scale:Nnn</code>	1251
<code>\box_set_dp:Nn</code> .	1131, 1197, 1294, 1318
<code>\box_set_ht:Nn</code> .	1132, 1196, 1295, 1317
<code>\box_set_wd:Nn</code> .	1133, 1195, 1296, 1316
<code>\box_use:N</code>	1300
<code>\box_use_drop:N</code> .	1144, 1198, 1275, 1319
<code>\box_wd:N</code> .	951, 1032, 1124, 1227, 1237
C	
clist commands:	
<code>\clist_const:Nn</code>	414
<code>\clist_map_function:NN</code>	857
<code>\clist_map_inline:Nn</code> .	423, 457, 469, 652
cs commands:	
<code>\cs_generate_variant:Nn</code>	


```

\__kernel_backend_literal_page:n
..... 28, 670, 694,
717, 726, 739, 748, 778, 808, 817, 830
\__kernel_backend_postscript:n .
..... 35, 1254, 1276, 1282, 1309
\__kernel_backend_shipout_-
literal:n ..... 39, 39, 525, 640
\__kernel_backend_shipout_-
literal_page:n ... 53, 53, 733, 823
\__kernel_backend_shipout_-
literal_pdf:n ..... 43, 43
\__kernel_kern:n .... 1411, 1419,
1422, 1451, 1649, 1657, 1660, 1686
\__kernel_pdf_name_from_unicode_-
e:n ..... 97, 103
\__kernel_pdf_object_id_indexed:nn
..... 1718, 1744, 1775, 1801
\__kernel_pdffdict_name:n ... 226,
227, 229, 484, 654, 836, 847, 852,
945, 966, 977, 982, 987, 992, 1027,
1047, 1057, 1062, 1067, 1072, 1221
\g__kernel_pdfmanagement_end_-
run_code_tl ..... 112, 119, 126
\g__kernel_pdfmanagement_-
thispage_shipout_code_tl 135, 141

L
lualua commands:
\lualua: ..... 201, 280, 331, 370
lua commands:
\lua_load_module:n ..... 929

M
mode commands:
\mode_leave_vertical: .... 1136, 1299

N
\NewSocketPlug ..... 930

P
pdf commands:
\pdf_activate_indexed_structure_-
destination: ..... 1341, 1348
\pdf_activate_structure_destination:
..... 1341, 1342
\l_pdf_current_structure_-
destination_tl ..... 1338,
1384, 1390, 1435, 1441, 1485, 1490,
1515, 1520, 1554, 1559, 1584, 1589,
1629, 1677, 1718, 1744, 1775, 1801
\pdf_object_if_exist:nTF .....
.. 1384, 1435, 1485, 1515, 1554, 1584
\pdf_object_new:n ..... 425, 471
\pdf_object_ref:n .....
..... 432, 489, 531, 591, 659,
677, 686, 757, 772, 838, 979, 984,
989, 994, 1059, 1064, 1069, 1074,
1150, 1157, 1164, 1172, 1390, 1441
\pdf_object_ref_indexed:nn 1629, 1677
\pdf_object_ref_last: .. 869, 876, 883
\pdf_object_unnamed_write:nn ...
..... 623, 721, 812, 868, 875, 882
\pdf_object_write ..... 486
\pdf_object_write:nnn . 459, 475, 1889
pdf internal commands:
\__pdf_backend:n .....
32, 171, 479, 487, 883, 1137, 1145,
1146, 1153, 1160, 1167, 1175, 1190,
1365, 1386, 1414, 1415, 1425, 1437,
1606, 1625, 1652, 1653, 1663, 1673
\__pdf_backend_action:nnnnnnn ..
..... 1947, 1975, 2003, 2031, 2070
\__pdf_backend_bdc:nn .....
..... 13, 511, 516, 520, 521,
551, 553, 554, 632, 634, 635, 729, 819
\__pdf_backend_bdc_contobj:nn ..
..... 520, 553, 621, 634, 719, 810
\__pdf_backend_bdc_contstream:nn
.... 521, 554, 627, 724, 729, 815, 819
\__pdf_backend_bdc_shipout:nn ..
..... 523, 642, 735, 825
\__pdf_backend_bdc_shipout_-
contstream:nn .....
..... 638, 642, 731, 735, 821, 825
\__pdf_backend_bdcobject:n ....
..... 13, 511,
533, 560, 596, 624, 691, 722, 775, 813
\__pdf_backend_bdcobject:nn ....
..... 13, 511, 529, 558, 577, 667, 745
\__pdf_backend_bmc:n .....
..... 13, 511, 541, 564, 615, 715, 806
\__pdf_backend_catalog_gput:nn .. 20
\__pdf_backend_destination:nn ..
..... 1344, 1350, 1356, 1359
\__pdf_backend_destination:nnnn
..... 1345, 1351, 1357, 1360
\__pdf_backend_emc: .....
..... 13, 511, 537, 562, 644, 737, 828
\__pdf_backend_goto:nnnnnnn ....
..... 1960, 1987, 2015, 2046, 2071
\__pdf_backend_indexed_structure_-
destination:nn .....
.. 1350, 1359, 1603, 1604, 1699, 1756
\__pdf_backend_indexed_structure_-
destination:nnnn .....
.. 1351, 1360, 1603, 1688, 1734, 1791
\__pdf_backend_indexed_structure_-
destination_aux:nnnn . 1645, 1690

```

```

\__pdf_backend_luastring:n ....
    158, 242, 251, 263, 264, 275, 290, 291
\__pdf_backend_metadata_stream:n
    ..... 1864, 1876, 1887
\g__pdf_backend_name_int .....
    ..... 88, 579, 582, 590,
    598, 601, 609, 669, 671, 676, 685,
    693, 695, 700, 709, 747, 749, 777, 779
\__pdf_backend_Names_gpush:nn ..
    ..... 866, 873, 880, 889, 893
\__pdf_backend_NamesEmbeddedFiles_-
    add:nn ..... 895, 896, 899, 911
\g__pdf_backend_object_int ....
    ..... 1112, 1115, 1213, 1216, 1261
\__pdf_backend_object_last: ....
    ..... 535, 610, 701, 710, 787, 802
\__pdf_backend_object_write:nn .
    ..... 1870, 1882
\__pdf_backend_omit_charset:n ..
    ..... 1892, 1893, 1896, 1902
\__pdf_backend_omit_cidset:n ...
    ..... 1937, 1938, 1941
\__pdf_backend_omit_info:n ....
    .. 1907, 1908, 1915, 1921, 1928, 1934
\__pdf_backend_Page_gput:nn ....
    ..... 6, 178, 188, 257, 318, 357, 393
\__pdf_backend_Page_gremove:n ..
    ..... 6, 178, 195, 271, 325, 364, 399
\g__pdf_backend_page_int ..... 88
\__pdf_backend_Page_primitive:n
    ..... 6, 178, 181, 234, 247,
    311, 336, 345, 350, 375, 384, 390, 411
\__pdf_backend_PageResources:n .
    ..... 477, 496, 504
\c__pdf_backend_PageResources_-
    clist .. 413, 423, 457, 469, 652, 858
\__pdf_backend_PageResources_-
    gpush:n .....
    ..... 13, 511, 545, 566, 650, 742, 842
\__pdf_backend_PageResources_-
    gpush_aux:n ..... 833, 859
\__pdf_backend_PageResources_-
    gput:nnn 422, 438, 449, 481, 497, 505
\__pdf_backend_PageResources_-
    obj_gpush: . 422, 455, 493, 501, 509
\__pdf_backend_Pages_primitive:n
    ..... 146, 147, 154, 163, 169, 175
\__pdf_backend_pdfmark:n .... 36,
    518, 531, 535, 539, 543, 901, 1950, 1963
\__pdf_backend_record_abspage:n
    ..... 65, 76, 208, 760, 790
\__pdf_backend_ref_abspage:n ...
    ..... 71, 77, 211, 763, 793

\g__pdf_backend_resourceid_int .
    ..... 88, 207, 208, 211, 751, 756,
    760, 763, 771, 781, 786, 790, 793, 801
\__pdf_backend_set_regression_-
    data: ..... 1815
\__pdf_backend_shipout_bdc:nn ..
    ..... 13, 511, 556
\__pdf_backend_structure_-
    destination:nn .....
    .. 1344, 1356, 1362, 1363, 1468, 1537
\__pdf_backend_structure_-
    destination:nnnn .....
    .. 1345, 1357, 1362, 1453, 1507, 1576
\__pdf_backend_structure_-
    destination_aux:nnnn . 1407, 1455
\__pdf_backend_ThisPage_gpush:n
    ..... 6, 178, 224, 300, 343, 382, 407
\__pdf_backend_ThisPage_gput:nn
    ..... 6, 178, 204, 283, 334, 373, 404
\g__pdf_backend_thispage_-
    shipout_tl ..... 6
\l__pdf_backend_tmpa_box .....
    ..... 82, 942, 951, 954, 957, 997,
    1024, 1032, 1035, 1038, 1077, 1188,
    1195, 1196, 1197, 1198, 1218, 1227,
    1230, 1233, 1237, 1242, 1247, 1251,
    1275, 1307, 1316, 1317, 1318, 1319
\l__pdf_backend_tmpb_box .....
    .... 86, 1252, 1294, 1295, 1296, 1300
\l__pdf_backend_xform_bool ....
    ..... 512, 672,
    696, 752, 782, 944, 1026, 1119, 1220
\__pdf_backend_xform_if_exist:n
    ..... 1329, 1335
\__pdf_backend_xform_new:nnnn ..
    .... 935, 936, 1018, 1106, 1203, 1211
\__pdf_backend_xform_ref:n ....
    ..... 935, 1010,
    1092, 1139, 1181, 1192, 1205, 1321
\l__pdf_backend_xform_tmpdp_tl .
    ..... 1209, 1245, 1259, 1266
\l__pdf_backend_xform_tmplt_tl .
    ..... 1210, 1240, 1264
\l__pdf_backend_xform_tmpwd_tl .
    ..... 1208, 1235, 1265
\__pdf_backend_xform_use:n ....
    .... 935, 1003, 1083, 1186, 1204, 1305
\__pdf_object_retrieve:n . 1869, 1880
\g__pdf_tmpa_prop .. 82, 226, 231, 236
\l__pdf_tmpa_tl .....
    ..... 82, 209, 213, 215, 218, 761,
    765, 767, 770, 791, 795, 797, 800, 803

```

pdfannot internal commands:	
__pdfannot_backend_link_begin:n	
.....	1460
__pdfannot_backend_link_-	
begin:nnnw	1530, 1599, 1810
__pdfannot_backend_link_begin_-	
goto:nnw	1346, 1352, 1358
__pdfannot_backend_link_begin_-	
structure_goto:nnw	1346, 1352,
1358, 1362, 1458, 1528, 1597, 1808	
__pdfannot_backend_link_off: ..	918
__pdfannot_backend_link_on: ...	922
pdfdict commands:	
\pdfdict_gput:nnn
. 190, 218, 320, 359, 395, 440, 451,	
499, 507, 674, 698, 754, 769, 784, 799	
\pdfdict_gremove:nn	197, 327, 366, 401
\pdfdict_if_exist:nTF ..	213, 765, 795
\pdfdict_item:nn	236, 838, 853
\pdfdict_new:n	215, 767, 797
\pdfdict_put:nnn	1858
\pdfdict_show:n	803
\pdfdict_use:n	346, 385, 461, 972, 1053
\pdfextension	931
\pdfliteral	2
pdfmanagement commands:	
\pdfmanagement_add:nnn	1818, 1821,
1824, 1825, 1828, 1836, 1844, 1849	
\pdfnames	20
\pdfomitinfodict	1917
\pdfpageref	3
\pdfrunninglinkoff	916, 920
\pdfrunninglinkon	924
\pdftrailerid	1846
pdfxform commands:	
\pdfxform_dp:n	1142, 1197, 1318
\pdfxform_ht:n	1141, 1196, 1317
\pdfxform_if_exist:n	1335
\pdfxform_wd:n	1140, 1195, 1316
prg commands:	
\prg_new_conditional:Npnn	1329
\prg_new_eq_conditional:NNn	1335
\prg_return_false:	1333
\prg_return_true:	1332
prop commands:	
\prop_count:N	966, 1047
\prop_gclear:N	945, 1027, 1221
\prop_gput:Nnn	231, 484
\prop_gset_eq:NN	226
\prop_if_empty:NTF	654, 835, 976,
981, 986, 991, 1056, 1061, 1066, 1071	
\prop_if_exist:NTF	227, 846
\prop_map_function:NN	236, 851
\prop_map_inline:Nn	229
\prop_new:N	83
property commands:	
\property_record:nn	68
\property_ref:nn	73
\ProvidesExplFile	1
R	
\relax	132, 1850
S	
scan commands:	
\scan_stop:	1007,
1089, 1484, 1504, 1514, 1525, 1553,	
1573, 1583, 1594, 1715, 1732, 1741,	
1749, 1772, 1789, 1798, 1806, 1845,	
1879, 1898, 1904, 1917, 1930, 1943	
\setbox	931
\special	2
str commands:	
\str_case:nnTF
1370, 1391, 1472, 1492, 1541, 1561,	
1611, 1630, 1703, 1720, 1760, 1777	
\str_convert_pdfname:n	99, 485
\str_if_eq:nnTF	1269, 1280
sys commands:	
\c_sys_engine_exec_str	1860
\sys_gset_rand_seed:n	1817
\c_sys_timestamp_str	1824, 1825, 1858
T	
tag commands:	
\tag_if_active:TF ..	1995, 2023, 2057
TeX and L ^A T _E X 2 _ε commands:	
\@bsphack	67
\@esphack	69
\@kernel@after@enddocument@afterlastpage
109, 110	
\@kernel@after@shipout@background
130, 133	
\@kernel@after@shipout@lastpage
116, 117, 123, 124	
\@kernel@before@shipout@background
132	
\g@addto@macro	132, 133
\special	2
tex commands:	
\tex_directlua:D
156, 259, 273, 426, 428, 441, 442	
\tex_global:D	149, 183, 844
\tex_immediate:D	959, 1040, 1868, 1880
\tex_latelua:D	249, 285, 302, 680, 704
\tex_luaescapestring:D	244
\tex_pdfcompresslevel:D	1879
\tex_pdfdest:D	1470, 1487,
1509, 1517, 1701, 1716, 1736, 1742	

<code>\tex_pdfextension:D</code>	46, 56, 876, 1539, 1556, 1578, 1586, 1758, 1773, 1793, 1799, 1868, 2006, 2018
<code>\tex_pdflastxform:D</code>	1000, 1080
<code>\tex_pdfliteral:D</code>	49, 59
<code>\tex_pdfnames:D</code>	869
<code>\tex_pdfobj:D</code>	1880
<code>\tex_pdfomitcharset:D</code>	1898
<code>\tex_pdfoutline:D</code>	1978, 1990
<code>\tex_pdfpageattr:D</code>	183
<code>\tex_pdfpageresources:D</code>	844
<code>\tex_pdfpagesattr:D</code>	149
<code>\tex_pdfrefxform:D</code>	1005, 1085
<code>\tex_pdfsuppressptexinfo:D</code> . . .	1845
<code>\tex_pdftexrevision:D</code>	1466, 1697, 1913
<code>\tex_pdftexversion:D</code>	1465, 1696, 1912
<code>\tex_pdfvariable:D</code>	
.	1850, 1851, 1904, 1930, 1943
<code>\tex_pdfxform:D</code>	959, 1040
<code>\tex_special:D</code>	
.	40, 165, 313, 352, 2034, 2049
<code>\tex_the:D</code>	
.	951, 954, 957, 1032, 1035, 1038, 1124, 1127, 1130, 1227, 1230, 1233
<code>\tex_unexpanded:D</code>	244
<code>\tex_vss:D</code>	1417, 1449, 1655, 1684
text commands:	
<code>\text_expand:n</code>	99, 105
tl commands:	
<code>\c_space_tl</code>	582, 590, 601, 609, 671, 695, 749, 779, 1140, 1141, 1142, 1261
<code>\tl_const:Nn</code>	
.	949, 952, 955, 1030, 1033, 1036, 1122, 1125, 1128, 1225, 1228, 1231
<code>\tl_gput_right:Nn</code>	110, 117, 124
<code>\tl_if_blank:nTF</code>	1954, 1967, 1981, 1993, 2009, 2021, 2042, 2064
<code>\tl_if_exist:NTF</code>	130
<code>\tl_new:N</code>	84, 1208, 1209, 1210, 1339
<code>\tl_set:Nn</code>	
.	209, 761, 791, 1235, 1240, 1245
<code>\tl_to_str:n</code>	950, 953, 956, 999, 1006, 1012, 1031, 1034, 1037, 1079, 1087, 1093, 1114, 1123, 1126, 1129, 1183, 1215, 1226, 1229, 1232, 1288, 1312, 1323, 1331, 1490, 1520, 1559, 1589
<code>\tl_use:N</code>	1259, 1264, 1265, 1266
U	
<code>\unvbox</code>	931
V	
<code>\vbox</code>	931
vbox commands:	
<code>\vbox_to_zero:n</code>	1409, 1420, 1647, 1658