

The l3backend-testphase package

Additional backend PDF features

L^AT_EX PDF management bundle

The L^AT_EX Project*

Version 0.96u, released 2025-07-15

1 l3backend-testphase Implementation

```
1 <drivers>\ProvidesExplFile
2 <*dvipdfmx>
3   {l3backend-testphase-dvipdfmx.def}{2025-07-15}{0.96s}
4   {LaTeX~PDF~management~bundle~backend~support: dvipdfmx}
5 </dvipdfmx>
6 <*dvips>
7   {l3backend-testphase-dvips.def}{2025-07-15}{0.96s}
8   {LaTeX~PDF~management~bundle~backend~support: dvips}
9 </dvips>
10 <*dvisvgm>
11   {l3backend-testphase-dvisvgm.def}{2025-07-15}{0.96s}
12   {LaTeX~PDF~management~bundle~backend~support: dvisvgm}
13 </dvisvgm>
14 <*luatex>
15   {l3backend-testphase-luatex.def}{2025-07-15}{0.96s}
16   {LaTeX~PDF~management~bundle~backend~support: PDF output (LuaTeX)}
17 </luatex>
18 <*pdftex>
19   {l3backend-testphase-pdftex.def}{2025-07-15}{0.96s}
20   {LaTeX~PDF~management~bundle~backend~support: PDF output (pdfTeX)}
21 </pdftex>
22 <*xdvipdfmx>
23   {l3backend-testphase-xetex.def}{2025-07-15}{0.96s}
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_tmpb_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-testphase.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 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

```

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     \prop_if_empty:cF
460     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/##1 } }
461     {
462       \pdf_object_write:nne
463       { __pdf/Page/Resources/##1 } { dict }
464       { \pdffdict_use:n { g__pdf_Core/Page/Resources/##1 } }
465     }
466   }
467 }
468 </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.

```

469 <*dvipdfmx | xdvipdfmx>
470 <xdvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-xetex}{after}{pdf}
471 <dvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-dvipdfmx}{after}{pdf}
472 %
473 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
474 {
475   \pdf_object_new:n { __pdf/Page/Resources/#1 }
476   \hook_gput_code:nnn
477   {shipout/firstpage}
478   {pdf}
479   {\pdf_object_write:nnn { __pdf/Page/Resources/#1 } { dict } {}}
480 }
481 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1
482 {
483   \__pdf_backend:n {put~@resources~<<#1>>}
484 }
485 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
486 {
487   % this is not used for output, but there is a test if the resource is empty
488   \prop_gput:cne { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/#1 } }
489   { \str_convert_pdfname:n {#2} }{ #3 }
490   %objects are not filled with \pdf_object_write as this is not additive!
491   \__pdf_backend:e
492   {
493     put~\pdf_object_ref:n {__pdf/Page/Resources/#1}<</#2~#3>>
494   }
495 }
496
497 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
498 </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.

```

499 <*dvips>
500 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
501 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
502 { %only for the show command TEST!!
503   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
504 }
505 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
506 </dvips>

```

dvipsvgm unneeded, or no-op

```

507 <*dvisvgm>
508 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
509 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
510 { %only for the show command TEST!!
511   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
512 }
513 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
514 </dvisvgm>

```

(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_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

```

...
515 <*drivers>
516 \bool_new:N \l__pdf_backend_xform_bool
517 </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>

```

518 <*dvips>
519 %
520 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
521 {
522   \__pdf_backend_pdfmark:n{/#1-<<#2>>~/BDC}
523 }

```

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

```

524 \cs_set_eq:NN \__pdf_backend_bdc_contobj:nn \__pdf_backend_bdc:nn
525 \cs_set_eq:NN \__pdf_backend_bdc_contstream:nn \__pdf_backend_bdc:nn
526
527 \cs_new_protected:Npn \__pdf_backend_bdc_shipout:ee #1 #2 % #1 eg. Span, #2: dict_content
528 {

```

```

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

```

```

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

```

```

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

```



```

691         )
692     }
693 }
694 }
695 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
696 {
697     \int_gincr:N \g__pdf_backend_name_int
698     \__kernel_backend_literal_page:e
699     { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
700     \bool_if:NTF \l__pdf_backend_xform_bool
701     {
702         \pdfdict_gput:nee %no handler needed
703         { g__pdf_Core/Xform/Resources/Properties }
704         { l3pdf\int_use:N\g__pdf_backend_name_int }
705         { \__pdf_backend_object_last: }
706     }
707     {
708         \exp_args:Ne \tex_latelua:D
709         {
710             ltx.pdf.Page_Resources_Properties_gput
711             (
712                 tex.count["g_shipout_readonly_int"],
713                 "l3pdf\int_use:N\g__pdf_backend_name_int",
714                 "\__pdf_backend_object_last:"
715             )
716         }
717     }
718 }
719 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
720 {
721     \__kernel_backend_literal_page:n { /#1~BMC }
722 }
723 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
724 {
725     \pdf_object_unnamed_write:nn { dict } { #2 }
726     \__pdf_backend_bdcobject:n { #1 }
727 }
728 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
729 {
730     \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
731 }
732
733 \cs_set_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn
734
735 \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
736 {
737     \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
738 }
739 \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
740
741 \cs_set_protected:Npn \__pdf_backend_emc:
742 {
743     \__kernel_backend_literal_page:n { EMC }
744 }

```

```

745
746 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
747 </luatex>

```

pdfflatex 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

```

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

```

```

796         {
797             \__pdf_backend_ref_abbrev: e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
798         }
799         \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
800         {
801             \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
802         }
803         \pdfdict_gput:nee
804         { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
805         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
806         { \__pdf_backend_object_last: }
807         %\pdfdict_show:n { g_backend_Page\l__pdf_tmpa_tl/Resources/Properties }
808     }
809 }
810 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
811 {
812     \__kernel_backend_literal_page:n { /#1~BMC }
813 }
814 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
815 {
816     \pdf_object_unnamed_write:nn { dict } { #2 }
817     \__pdf_backend_bdcobject:n { #1 }
818 }
819 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
820 {
821     \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
822 }

```

We use by default the direct BDC.

```

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

```

```

849     {
850         \prop_if_exist:cT
851         { \__kernel_pdffdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
852         {
853             /Properties~
854             <<
855                 \prop_map_function:cN
856                 { \__kernel_pdffdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
857                 \pdffdict_item:ne
858             >>
859         }
860         %% add ExtGState etc
861         \clist_map_function:NN
862         \c__pdf_backend_PageResources_clist
863         \__pdf_backend_PageResources_gpush_aux:n
864     }
865 }
866
867 </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.

```

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

```

```

889 </dvipdfmx | xdvipdfmx>
890
891 %dvips: noop
892 <*dvips>
893 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
894 </dvips>
895 %dvisvgm: noop
896 <*dvisvgm>
897 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
898 </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.

```

899 <*pdftex | luatex | dvipdfmx | xdvipdfmx>
900 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2 {}
901 </pdftex | luatex | dvipdfmx | xdvipdfmx>
902 <*dvips>
903 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
904 {
905     \__pdf_backend_pdfmark:e
906     {
907         /Name~#1~
908         /FS~#2~
909         /EMBED
910     }
911 }
912 </dvips>
913 <*dvisvgm>
914 %no op. Or is there any sensible use for it?
915 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
916 {}
917
918 </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.

```

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

```

```

929     }
930   }
931 </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.

```

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

```

```

966 ~ resources ~
967 {
968   #3
969   \int_compare:nNnT
970     { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
971     >
972     { 0 }
973     {
974       /Properties~
975       <<
976       \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
977       >>
978     }
979
980   \prop_if_empty:cF
981     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
982     {
983       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
984     }
985   \prop_if_empty:cF
986     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
987     {
988       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
989     }
990   \prop_if_empty:cF
991     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
992     {
993       /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
994     }
995   \prop_if_empty:cF
996     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
997     {
998       /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
999     }
1000   }
1001   \l__pdf_backend_tmpa_box
1002   \int_const:cn
1003     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1004     { \tex_pdflastxform:D }
1005   }
1006
1007   \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1008     {
1009       \tex_pdfrefxform:D
1010       \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1011       \scan_stop:
1012     }
1013
1014   \cs_new:Npn \__pdf_backend_xform_ref:n #1
1015     {
1016       \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
1017     }
1018   </pdfTeX>
1019   <*luatex>

```

```

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

```



```

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

```

```

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

```

```

1182
1183
1184
1185 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1186 {
1187     @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1188 }
1189
1190 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1191 {
1192     \hbox_set:Nn \l__pdf_backend_tmpa_box
1193     {
1194         \__pdf_backend:e
1195         {
1196             uxobj~ \__pdf_backend_xform_ref:n { #1 }
1197         }
1198     }
1199     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1200     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1201     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1202     \box_use_drop:N \l__pdf_backend_tmpa_box
1203 }
1204 </dviptfm> | xdvipdfmx>
1205 <*dvisvgm>
1206 % unclear what it should do!!
1207 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}
1208 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
1209 \cs_new:Npn \__pdf_backend_xform_ref:n {}
1210 </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

```

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

```

```

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

```

```

1285     {
1286         \__kernel_backend_postscript:e
1287         {
1288             product~(Ghostscript)~search~
1289             {
1290                 pop~pop~pop~
1291                 mark~
1292                 { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1293                 ~<<#2>>~/PUT~pdfmark
1294             }{pop}ifelse
1295         }
1296     }
1297 }
1298 \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1299 \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1300 \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1301 \hook_gput_code:nnn {begindocument/end}{pdfxform}
1302 {
1303     \mode_leave_vertical:
1304     \box_use:N\l__pdf_backend_tmpb_box
1305 }
1306 }
1307
1308
1309 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1310 {
1311     \hbox_set:Nn \l__pdf_backend_tmpa_box
1312     {
1313         \__kernel_backend_postscript:e
1314         {
1315             gsave~currentpoint~translate~1~-1~scale~
1316             mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }~
1317             /SP~pdfmark ~ grestore
1318         }
1319     }
1320     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1321     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1322     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1323     \box_use_drop:N \l__pdf_backend_tmpa_box
1324 }
1325 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1326 {
1327     { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1328 }
1329
1330 </dvips>
1331 <*drivers>
1332 %% all
1333 \prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
1334 {
1335     \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1336     { \prg_return_true: }
1337     { \prg_return_false: }
1338 }

```

```

1339 \prg_new_eq_conditional:NNn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
1340 { TF , T , F , p }
1341 </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 luatex it was only possible to create a restricted type which used only the “Fit” mode. Starting with T_EXlive 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
1342 <*drivers>
1343 \tl_new:N \l_pdf_current_structure_destination_tl
1344 </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:
1345 <*drivers>
1346 \cs_new_protected:Npn \pdf_activate_structure_destination:
1347 {
1348   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_structure_destination:nn
1349   \cs_gset_eq:NN \__pdf_backend_destination:nnnn \__pdf_backend_structure_destination:nnnn
1350   \cs_gset_eq:NN \__pdfannot_backend_link_begin_goto:nnw \__pdfannot_backend_link_begin_goto:nnw
1351 }
1352 \cs_new_protected:Npn \pdf_activate_indexed_structure_destination:
1353 {
1354   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_indexed_structure_destination:nn
1355   \cs_gset_eq:NN \__pdf_backend_destination:nnnn \__pdf_backend_indexed_structure_destination:nnnn
1356   \cs_gset_eq:NN \__pdfannot_backend_link_begin_goto:nnw \__pdfannot_backend_link_begin_goto:nnw
1357 }
1358 </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.

```

1359 <*drivers>
1360 \cs_set_eq:NN \__pdf_backend_structure_destination:nn \__pdf_backend_destination:nn
1361 \cs_set_eq:NN \__pdf_backend_structure_destination:nnnn \__pdf_backend_destination:nnnn
1362 \cs_set_eq:NN \__pdfannot_backend_link_begin_structure_goto:nnw \__pdfannot_backend_link_begin_goto:nnw
1363 \cs_set_eq:NN \__pdf_backend_indexed_structure_destination:nn \__pdf_backend_destination:nn
1364 \cs_set_eq:NN \__pdf_backend_indexed_structure_destination:nnnn \__pdf_backend_destination:nnnn
1365 </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:

```

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

```

```

1383         { fitr } { /Fit }
1384     }
1385     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1386 ]
1387 }

```

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.

```

1388 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1389 {
1390     \__pdf_backend:e
1391     {
1392         obj ~ @pdf.SDest.\exp_not:n{#1}
1393         [
1394             \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1395             \str_case:nnF {#2}
1396             {
1397                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1398                 { fit } { /Fit }
1399                 { fitb } { /FitB }
1400                 { fitbh } { /FitBH }
1401                 { fitbv } { /FitBV ~ @xpos }
1402                 { fith } { /FitH ~ @ypos }
1403                 { fitv } { /FitV ~ @xpos }
1404                 { fitr } { /Fit }
1405             }
1406             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1407         ]
1408     }
1409 }
1410 }

```

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

```

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

```



```

1431         dest ~ (#2)
1432     [
1433         @thispage
1434         /FitR ~
1435         @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1436         @xpos ~ @ypos
1437     ]
1438 }

```

Here we add the structure destination to the same box

```

1439     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1440     {
1441         \__pdf_backend:e
1442         {
1443             obj ~ @pdf.SDest.\exp_not:n{#2}
1444             [
1445                 \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1446                 /FitR ~
1447                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1448                 @xpos ~ @ypos
1449             ]
1450         }
1451     }
1452 }
1453 \tex_vss:D
1454 }
1455 \__kernel_kern:n { -#1 }
1456 }

```

And now we redefine the destination command:

```

1457 \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1458 {
1459     \exp_args:Ne \__pdf_backend_structure_destination_aux:nnnn
1460     { \dim_eval:n {#2} } {#1} {#3} {#4}
1461 }

```

At last the goto link.

```

1462 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1463 {
1464     \__pdfannot_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD~@pdf.S
1465 }
1466 </xdvipdfmx|dvipdfmx>

```

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

```

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

```

```

1479         { fit } { fit }
1480         { fitb } { fitb }
1481         { fitbh } { fitbh }
1482         { fitbv } { fitbv }
1483         { fith } { fith }
1484         { fitv } { fitv }
1485         { fitr } { fitr }
1486     }
1487     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1488     \scan_stop:
1489     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1490     {
1491         \tex_pdfdest:D
1492         struct~
1493         \int_use:c
1494         { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl}
1495         name {#1}
1496         \str_case:nnF {#2}
1497         {
1498             { xyz } { xyz }
1499             { fit } { fit }
1500             { fitb } { fitb }
1501             { fitbh } { fitbh }
1502             { fitbv } { fitbv }
1503             { fith } { fith }
1504             { fitv } { fitv }
1505             { fitr } { fitr }
1506         }
1507         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1508         \scan_stop:
1509     }
1510 }
1511 \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1512 {
1513     \tex_pdfdest:D
1514     name {#1}
1515     fitr ~
1516     width \dim_eval:n {#2} ~
1517     height \dim_eval:n {#3} ~
1518     depth \dim_eval:n {#4} \scan_stop:
1519     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1520     {
1521         \tex_pdfdest:D
1522         struct~
1523         \int_use:c
1524         { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl}
1525         name {#1}
1526         fitr ~
1527         width \dim_eval:n {#2} ~
1528         height \dim_eval:n {#3} ~
1529         depth \dim_eval:n {#4} \scan_stop:
1530     }
1531 }
1532 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nw #1#2

```

```

1533     {
1534         \__pdfannot_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1535     }
1536 }
1537 </pdfTeX>

```

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

```

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

```

```

1586     height \dim_eval:n {#3} ~
1587     depth \dim_eval:n {#4} \scan_stop:
1588 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1589 {
1590     \tex_pdfextension:D dest
1591     struct~
1592     \int_use:c
1593     { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl}
1594     name {#1}
1595     fitr ~
1596     width \dim_eval:n {#2} ~
1597     height \dim_eval:n {#3} ~
1598     depth \dim_eval:n {#4} \scan_stop:
1599 }
1600 }
1601 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1602 {
1603     \__pdfannot_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1604 }
1605 }
1606 </luatex>

```

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

df_backend_indexed_structure_destination:nn
_backend_indexed_structure_destination:nnnn

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:

```

1607 <*xdvipdfmx|dvipdfmx>
1608 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nn #1#2
1609 {
1610     \__pdf_backend:e
1611     {
1612         dest ~ ( \exp_not:n {#1} )
1613         [
1614             @thispage
1615             \str_case:nnF {#2}
1616             {
1617                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1618                 { fit } { /Fit }
1619                 { fitb } { /FitB }
1620                 { fitbh } { /FitBH }
1621                 { fitbv } { /FitBV ~ @xpos }
1622                 { fith } { /FitH ~ @ypos }
1623                 { fitv } { /FitV ~ @xpos }
1624                 { fitr } { /Fit }
1625             }
1626             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1627         ]
1628     }

```

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.

```

1629 \__pdf_backend:e
1630 {
1631   obj ~ @pdf.SDest.\exp_not:n{#1}
1632   [
1633     \exp_after:wN \pdf_object_ref_indexed:nn \l_pdf_current_structure_destination_t
1634     \str_case:nnF {#2}
1635     {
1636       { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1637       { fit } { /Fit }
1638       { fitb } { /FitB }
1639       { fitbh } { /FitBH }
1640       { fitbv } { /FitBV ~ @xpos }
1641       { fith } { /FitH ~ @ypos }
1642       { fitv } { /FitV ~ @xpos }
1643       { fitr } { /Fit }
1644     }
1645     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1646   ]
1647 }
1648 }

```

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

```

1649 \cs_new_protected:Npn \__pdf_backend_indexed_structure_destination_aux:nnnn #1#2#3#4
1650 {
1651   \vbox_to_zero:n
1652   {
1653     \__kernel_kern:n {#4}
1654     \hbox:n
1655     {
1656       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1657       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1658     }
1659     \tex_vss:D
1660   }
1661   \__kernel_kern:n {#1}
1662   \vbox_to_zero:n
1663   {
1664     \__kernel_kern:n { -#3 }
1665     \hbox:n
1666     {
1667       \__pdf_backend:n
1668       {
1669         dest ~ (#2)
1670         [
1671           @thispage
1672           /FitR ~
1673           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1674           @xpos ~ @ypos
1675         ]
1676       }

```

Here we add the structure destination to the same box

```

1677 \__pdf_backend:e
1678 {

```

```

1679         obj ~ @pdf.SDest.\exp_not:n{#2}
1680     [
1681         \exp_after:wN \pdf_object_ref_indexed:nn \l_pdf_current_structure_destin
1682         /FitR ~
1683         @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1684         @xpos ~ @ypos
1685     ]
1686 }
1687 }
1688 \tex_vss:D
1689 }
1690 \__kernel_kern:n { -#1 }
1691 }

```

And now we redefine the destination command:

```

1692 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1693 {
1694     \exp_args:Ne \__pdf_backend_indexed_structure_destination_aux:nnnn
1695     { \dim_eval:n {#2} } {#1} {#3} {#4}
1696 }
1697 </xdvipdfmx|dvipdfmx>

```

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

```

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

```

```

1730         { fitbv } { fitbv }
1731         { fith } { fith }
1732         { fitv } { fitv }
1733         { fitr } { fitr }
1734     }
1735     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1736     \scan_stop:
1737 }
1738 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1739 {
1740     \tex_pdfdest:D
1741     name {#1}
1742     fitr ~
1743     width \dim_eval:n {#2} ~
1744     height \dim_eval:n {#3} ~
1745     depth \dim_eval:n {#4} \scan_stop:
1746     \tex_pdfdest:D
1747     struct~
1748     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_destinati
1749     name {#1}
1750     fitr ~
1751     width \dim_eval:n {#2} ~
1752     height \dim_eval:n {#3} ~
1753     depth \dim_eval:n {#4} \scan_stop:
1754 }
1755 }
1756 </pdfTeX>

```

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

```

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

```

```

1783         { xyz }   { xyz }
1784         { fit }   { fit }
1785         { fitb }  { fitb }
1786         { fitbh } { fitbh }
1787         { fitbv } { fitbv }
1788         { fith }  { fith }
1789         { fitv }  { fitv }
1790         { fitr }  { fitr }
1791     }
1792     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1793     \scan_stop:
1794 }
1795 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1796 {
1797     \tex_pdfextension:D dest
1798     name {#1}
1799     fitr ~
1800     width \dim_eval:n {#2} ~
1801     height \dim_eval:n {#3} ~
1802     depth \dim_eval:n {#4} \scan_stop:
1803     \tex_pdfextension:D dest
1804     struct~
1805     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_destination:nn
1806     name {#1}
1807     fitr ~
1808     width \dim_eval:n {#2} ~
1809     height \dim_eval:n {#3} ~
1810     depth \dim_eval:n {#4} \scan_stop:
1811 }
1812 \cs_set_protected:Npn \__pdfannot_backend_link_begin_structure_goto:nnw #1#2
1813 {
1814     \__pdfannot_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1815 }
1816 }
1817 </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

```

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

```



```

1828     \pdfmanagement_add:nne{Info}{CreationDate}{(\c_sys_timestamp_str)}
1829     \pdfmanagement_add:nne{Info}{ModDate}{(\c_sys_timestamp_str)}
1830   </dvips>
1831   <*dvipdfmx>
1832     \pdfmanagement_add:nnn{Info}{Producer}{(dvipdfmx)}
1833     \__kernel_backend_literal:e
1834       {pdf:trailerid [~
1835         <00112233445566778899aabbccddeeff>~
1836         <00112233445566778899aabbccddeeff>~
1837       ]}
1838   </dvipdfmx>
1839   <*xdvipdfmx>
1840     \pdfmanagement_add:nnn{Info}{Producer}{(xetex)}
1841     \__kernel_backend_literal:e
1842       {pdf:trailerid [~
1843         <00112233445566778899aabbccddeeff>~
1844         <00112233445566778899aabbccddeeff>~
1845       ]}
1846   </xdvipdfmx>
1847   <*pdftex>
1848     \pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX)}
1849     \tex_pdfsuppressptexinfo:D 7 \scan_stop:
1850     \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
1851   </pdftex>
1852   <*luatex>
1853     \pdfmanagement_add:nnn{Info}{Producer}{(LuaTeX)}
1854     \tex_pdfvariable:D suppressoptionalinfo 7\relax
1855     \tex_pdfvariable:D trailerid
1856     {[~
1857       <2350CAD05F8A7AF0AA4058486855344F>~
1858       <2350CAD05F8A7AF0AA4058486855344F>~
1859     ]}
1860   </luatex>

```

Embedded files should also have a fix date.

```

1861   <*drivers>
1862     \pdfdict_put:nne {l_pdffile/Params} {ModDate}{(\c_sys_timestamp_str)}
1863     \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-
1864       8c19-6237d832fc80}
1865     \AddToDocumentProperties[hyperref]{pdfproducer}{\c_sys_engine_exec_str-NN.NN.NN}
1866   </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)dvipdfmx 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 an unnamed object so should be referenced directly with `\pdf_object_ref_last:`

```

1867 <*/luatex>
1868 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1869 {
1870   \tex_immediate:D \tex_pdfextension:D obj ~uncompressed~
1871   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1872 }
1873 </luatex>
1874 <*pdftex>
1875 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1876 {
1877   \group_begin:
1878   \tex_pdfcompresslevel:D 0 \scan_stop:
1879   \tex_immediate:D \tex_pdfobj:D
1880   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1881   \group_end:
1882 }
1883 </pdftex>
1884 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1885 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1886 {
1887   \pdf_object_unnamed_write:nn {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
1888 }
1889 </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.

```

1890 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1891 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 {} % #1 number
1892 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1893 <*pdftex>
1894 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 % #1 number
1895 {
1896   \tex_pdfomitcharset:D = #1 \scan_stop:
1897 }
1898 </pdftex>
1899 <*/luatex>
1900 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 % #1 number
1901 {
1902   \tex_pdfvariable:D omitcharset = #1 \scan_stop:
1903 }
1904 </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.

```

1905 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1906 \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {} % #1 number
1907 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>

```

```

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

```

1935 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm | pdfTeX>
1936 \cs_new_protected:Npn \__pdf_backend_omit_cidset:n #1 {}%#1 number
1937 </xdvipdfmx | dvipdfmx | dvips | dvisvgm | pdfTeX>
1938 <*luatex>
1939 \cs_new_protected:Npn \__pdf_backend_omit_cidset:n #1 {%#1 number
1940   {
1941     \tex_pdfvariable:D omitcidset = #1 \scan_stop:
1942   }
1943 </luatex>

```

(End of definition for __pdf_backend_omit_cidset:n.)

1.15 lua code for lualatex

```

1944 <*lua>
1945 ltx= ltx or {}
1946 ltx.__pdf = ltx.__pdf or {}
1947 ltx.__pdf.Page = ltx.__pdf.Page or {}
1948 ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
1949 ltx.__pdf.Page.Resources = ltx.__pdf.Page.Resources or {}
1950 ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
1951 ltx.__pdf.Page.Resources.List={"ExtGState","ColorSpace","Pattern","Shading"}

```

```

1952 ltx.__pdf.object = ltx.__pdf.object or {}
1953
1954 ltx.pdf= ltx.pdf or {} -- for "public" functions
1955
1956 local __pdf = ltx.__pdf
1957 local pdf = pdf
1958
1959 local function __pdf_backend_Page_gput (name,value)
1960 __pdf.Page.dflt[name]=value
1961 end
1962
1963 local function __pdf_backend_Page_gremove (name)
1964 __pdf.Page.dflt[name]=nil
1965 end
1966
1967 local function __pdf_backend_Page_gclear ()
1968 __pdf.Page.dflt={}
1969 end
1970
1971 local function __pdf_backend_ThisPage_gput (page,name,value)
1972 __pdf.Page[page] = __pdf.Page[page] or {}
1973 __pdf.Page[page][name]=value
1974 end
1975
1976 local function __pdf_backend_ThisPage_gpush (page)
1977 local token=""
1978 local t = {}
1979 local tkeys= {}
1980 for name,value in pairs(__pdf.Page.dflt) do
1981 t[name]=value
1982 end
1983 if __pdf.Page[page] then
1984 for name,value in pairs(__pdf.Page[page]) do
1985 t[name] = value
1986 end
1987 end
1988 -- sort the table to get reliable test files.
1989 for name,value in pairs(t) do
1990 table.insert(tkeys,name)
1991 end
1992 table.sort(tkeys)
1993 for _,name in ipairs(tkeys) do
1994 token = token .. "/"..name.." "..t[name]
1995 end
1996 return token
1997 end
1998
1999 function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly_
2000 __pdf_backend_ThisPage_gput (page,name,value)
2001 end
2002
2003 function ltx.__pdf.backend_ThisPage_gpush (page)
2004 pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
2005 end

```

```

2006
2007 function ltx.__pdf.backend_Page_gput (name,value)
2008   __pdf_backend_Page_gput (name,value)
2009 end
2010
2011 function ltx.__pdf.backend_Page_gremove (name)
2012   __pdf_backend_Page_gremove (name)
2013 end
2014
2015 function ltx.__pdf.backend_Page_gclear ()
2016   __pdf_backend_Page_gclear ()
2017 end
2018
2019
2020 local Properties = ltx.__pdf.Page.Resources.Properties
2021 local ResourceList= ltx.__pdf.Page.Resources.List
2022 local function __pdf_backend_PageResources_gpush (page)
2023   local token=""
2024   if Properties[page] then
2025     -- we sort the table, so that the pdf test works
2026     local t = {}
2027     for name,value in pairs (Properties[page]) do
2028       table.insert (t,name)
2029     end
2030     table.sort (t)
2031     for _,name in ipairs(t) do
2032       token = token .. "/"..name.." " .. Properties[page][name]
2033     end
2034     token = "/Properties <<"..token..">>"
2035   end
2036   for i,name in ipairs(ResourceList) do
2037     if ltx.__pdf.Page.Resources[name] then
2038       token = token .. "/"..name.." " ..ltx.pdf.object_ref("__pdf/Page/Resources/"..name)
2039     end
2040   end
2041   return token
2042 end
2043
2044 -- the function is public, as I probably need it in tagpdf too ...
2045 function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
2046   Properties[page] = Properties[page] or {}
2047   Properties[page][name]=value
2048   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
2049 end
2050
2051 function ltx.pdf.Page_Resources_gpush(page)
2052   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
2053 end
2054
2055 function ltx.pdf.object_ref (objname)
2056   if ltx.__pdf.object[objname] then
2057     local ref= ltx.__pdf.object[objname]
2058     return ref
2059   else

```

```

2060     return "false"
2061 end
2062 end
2063  $\langle$ /lua $\rangle$ 

```

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	
\backslash AddToDocumentProperties	1863, 1864
\backslash AddToHook	1825
\backslash AssignSocketPlug	937
B	
bool commands:	
\backslash bool_if:NTF	676, 700, 756, 786
\backslash bool_lazy_and:nnTF	1468, 1699, 1909
\backslash bool_new:N	516
\backslash bool_set_true:N	948, 1030, 1123, 1224
box commands:	
\backslash box_dp:N	961, 1042, 1134, 1237, 1251
\backslash box_ht:N	958, 1039, 1131, 1234, 1246
\backslash box_new:N	85, 86, 1120
\backslash box_scale:Nnn	1255
\backslash box_set_dp:Nn	1135, 1201, 1298, 1322
\backslash box_set_ht:Nn	1136, 1200, 1299, 1321
\backslash box_set_wd:Nn	1137, 1199, 1300, 1320
\backslash box_use:N	1304
\backslash box_use_drop:N	1148, 1202, 1279, 1323
\backslash box_wd:N	955, 1036, 1128, 1231, 1241
C	
clist commands:	
\backslash clist_const:Nn	414
\backslash clist_map_function:NN	861
\backslash clist_map_inline:Nn	423, 457, 473, 656
cs commands:	
\backslash cs_generate_variant:Nn	28, 31, 32, 35, 36, 76, 77, 411
\backslash cs_gset_eq:NN	638, 1348, 1349, 1350, 1354, 1355, 1356
\backslash cs_if_exist:NTF	426, 920
\backslash cs_new:Npn	71, 97, 103, 242, 837, 1014, 1096, 1185, 1209, 1325
\backslash cs_new_protected:Npn	39, 43, 53, 65, 147, 154, 169, 175, 181, 188, 195, 204, 224, 247, 257, 271, 283, 300, 311, 318, 325, 334, 343, 350, 357, 364, 373, 382, 390, 393, 399, 404, 407, 438, 449, 455, 481, 485, 497, 500, 501, 505, 508, 509, 513, 527, 549, 570, 654, 746, 846, 870, 877, 884, 893, 897, 900, 903, 915, 940, 1007, 1022, 1087, 1110, 1190, 1207, 1208, 1215, 1309, 1346, 1352, 1411, 1649, 1819, 1868, 1875, 1885, 1891, 1894, 1900, 1906, 1913, 1919, 1926, 1932, 1936, 1939
\backslash cs_new_protected:Npx	163
\backslash cs_set_eq:NN	524, 525, 557, 558, 646, 733, 739, 823, 829, 1360, 1361, 1362, 1363, 1364
\backslash cs_set_protected:Npn	520, 533, 537, 541, 545, 555, 560, 562, 564, 566, 568, 581, 600, 619, 625, 631, 636, 642, 648, 671, 695, 719, 723, 728, 735, 741, 749, 779, 810, 814, 819, 825, 832, 922, 926, 1367, 1457, 1462, 1472, 1511, 1532, 1541, 1580, 1601, 1608, 1692, 1703, 1738, 1760, 1795, 1812
D	
dim commands:	
\backslash dim_eval:n	1460, 1516, 1517, 1518, 1527, 1528, 1529, 1585, 1586, 1587, 1596, 1597, 1598, 1695, 1743, 1744, 1745, 1751, 1752, 1753, 1800, 1801, 1802, 1808, 1809, 1810
\backslash dim_to_decimal_in_sp:n	1241, 1246, 1251
\backslash c_zero_dim	1135, 1136, 1137, 1298, 1299, 1300
\backslash directlua	94, 1539, 1758, 1924
E	
exp commands:	
\backslash exp_after:wN	1633, 1681, 1722, 1748, 1779, 1805
\backslash exp_args:Ne	684, 708, 1388, 1394, 1439, 1445, 1459, 1489, 1494, 1519, 1524, 1558, 1563, 1588, 1593, 1694

<code>\exp_args:NNe</code>	848	<code>__kernel_backend_shipout_-</code>	
<code>\exp_not:n</code>	605, 699, 783, 1371, 1392, 1443, 1612, 1631, 1679	<code>literal:n</code>	39, 39, 529, 644
F			
<code>\footins</code>	935	<code>__kernel_backend_shipout_-</code>	
fp commands:		<code>literal_page:n</code>	53, 53, 737, 827
<code>\fp_eval:n</code>		<code>__kernel_backend_shipout_-</code>	
	1385, 1406, 1487, 1507, 1556, 1576, 1626, 1645, 1718, 1735, 1775, 1792	<code>literal_pdf:n</code>	43, 43
G			
group commands:		<code>__kernel_kern:n</code>	1415, 1423, 1426, 1455, 1653, 1661, 1664, 1690
<code>\group_begin:</code>	1877	<code>__kernel_pdf_name_from_unicode_-</code>	
<code>\group_end:</code>	1881	<code>e:n</code>	97, 103
H			
hbox commands:		<code>__kernel_pdf_object_id_indexed:nn</code>	
<code>\hbox:n</code>	1416, 1427, 1654, 1665		1722, 1748, 1779, 1805
<code>\hbox_gset:Nn</code>	1121	<code>__kernel_pdftdict_name:n</code>	226, 227, 229, 460, 488, 658, 840, 851, 856, 949, 970, 981, 986, 991, 996, 1031, 1051, 1061, 1066, 1071, 1076, 1225
<code>\hbox_set:Nn</code>		<code>\g__kernel_pdfmanagement_end_-</code>	
	946, 1028, 1192, 1222, 1256, 1311	<code>run_code_tl</code>	112, 119, 126
hook commands:		<code>\g__kernel_pdfmanagement_-</code>	
<code>\hook_gput_code:nnn</code>	139, 476, 1301	<code>thispage_shipout_code_tl</code>	135, 141
<code>\hook_gput_next_code:nn</code>	1138	L	
<code>\hook_gset_rule:nnnn</code>	470, 471	lathlua commands:	
I		<code>\lathlua:</code>	201, 280, 331, 370
int commands:		lua commands:	
<code>\int_compare:nNnTF</code>		<code>\lua_load_module:n</code>	933
	969, 1050, 1539, 1758, 1924	M	
<code>\int_compare_p:nNn</code>		mode commands:	
	1469, 1470, 1700, 1701, 1910, 1911	<code>\mode_leave_vertical:</code>	1140, 1303
<code>\int_const:Nn</code>	1002, 1082, 1117, 1218	N	
<code>\int_gincr:N</code>	207, 583, 602, 673, 697, 751, 755, 781, 785, 1116, 1217	<code>\NewSocketPlug</code>	934
<code>\int_if_exist:NTF</code>	1335	P	
<code>\int_new:N</code>	89, 90, 91	pdf commands:	
<code>\int_use:N</code>	208, 211, 586, 594, 605, 613, 675, 680, 689, 699, 704, 713, 753, 760, 764, 767, 775, 783, 790, 794, 797, 805, 1010, 1016, 1089, 1097, 1187, 1265, 1292, 1316, 1327, 1493, 1523, 1562, 1592	<code>\pdf_activate_indexed_structure_-</code>	
K		<code>destination:</code>	1345, 1352
kernel internal commands:		<code>\pdf_activate_structure_destination:</code>	
<code>__kernel_backend_literal:n</code>			1345, 1346
	31, 80, 584, 588, 603, 607, 621, 633, 650, 660, 1826, 1827, 1833, 1841	<code>\l_pdf_current_structure_-</code>	
<code>__kernel_backend_literal_page:n</code>		<code>destination_tl</code>	1342, 1388, 1394, 1439, 1445, 1489, 1494, 1519, 1524, 1558, 1563, 1588, 1593, 1633, 1681, 1722, 1748, 1779, 1805
	28, 674, 698, 721, 730, 743, 752, 782, 812, 821, 834	<code>\pdf_object_if_exist:nTF</code>	
<code>__kernel_backend_postscript:n</code>	35, 1258, 1280, 1286, 1313		1388, 1439, 1489, 1519, 1558, 1588
		<code>\pdf_object_new:n</code>	425, 475
		<code>\pdf_object_ref:n</code>	
			432, 493, 535, 595, 663, 681, 690, 761, 776, 842, 983, 988, 993, 998, 1063, 1068, 1073, 1078, 1154, 1161, 1168, 1176, 1394, 1445
		<code>\pdf_object_ref_indexed:nn</code>	1633, 1681
		<code>\pdf_object_ref_last:</code> . .	873, 880, 887

```

\pdf_object_unnamed_write:nn ...
... 627, 725, 816, 872, 879, 886, 1887
\pdf_object_write ..... 490
\pdf_object_write:nnn ..... 462, 479
pdf internal commands:
\_pdf_backend:n .....
32, 171, 483, 491, 887, 1141, 1149,
1150, 1157, 1164, 1171, 1179, 1194,
1369, 1390, 1418, 1419, 1429, 1441,
1610, 1629, 1656, 1657, 1667, 1677
\_pdf_backend_bdc:nn .....
..... 13, 515, 520, 524, 525,
555, 557, 558, 636, 638, 639, 733, 823
\_pdf_backend_bdc_contobj:nn ..
..... 524, 557, 625, 638, 723, 814
\_pdf_backend_bdc_contstream:nn
.... 525, 558, 631, 728, 733, 819, 823
\_pdf_backend_bdc_shipout:nn ..
..... 527, 646, 739, 829
\_pdf_backend_bdc_shipout_-
contstream:nn .....
..... 642, 646, 735, 739, 825, 829
\_pdf_backend_bdcobject:n ....
..... 13, 515,
537, 564, 600, 628, 695, 726, 779, 817
\_pdf_backend_bdcobject:nn ....
..... 13, 515, 533, 562, 581, 671, 749
\_pdf_backend_bmc:n .....
..... 13, 515, 545, 568, 619, 719, 810
\_pdf_backend_catalog_gput:nn .. 20
\_pdf_backend_destination:nn ..
..... 1348, 1354, 1360, 1363
\_pdf_backend_destination:nnnn
..... 1349, 1355, 1361, 1364
\_pdf_backend_emc: .....
..... 13, 515, 541, 566, 648, 741, 832
\_pdf_backend_indexed_structure_-
destination:nn .....
.. 1354, 1363, 1607, 1608, 1703, 1760
\_pdf_backend_indexed_structure_-
destination:nnnn .....
.. 1355, 1364, 1607, 1692, 1738, 1795
\_pdf_backend_indexed_structure_-
destination_aux:nnnn .. 1649, 1694
\_pdf_backend_luastring:n ....
158, 242, 251, 263, 264, 275, 290, 291
\_pdf_backend_metadata_stream:n
..... 1868, 1875, 1885
\g_pdf_backend_name_int .....
..... 88, 583, 586, 594,
602, 605, 613, 673, 675, 680, 689,
697, 699, 704, 713, 751, 753, 781, 783
\_pdf_backend_Names_gpsh:n ..
..... 870, 877, 884, 893, 897
\_pdf_backend_NamesEmbeddedFiles_-
add:nn ..... 899, 900, 903, 915
\g_pdf_backend_object_int ....
..... 1116, 1119, 1217, 1220, 1265
\_pdf_backend_object_last: ....
..... 539, 614, 705, 714, 791, 806
\_pdf_backend_object_write:nn .
..... 1871, 1880
\_pdf_backend_omit_charset:n ..
..... 1890, 1891, 1894, 1900
\_pdf_backend_omit_cidset:n ...
..... 1935, 1936, 1939
\_pdf_backend_omit_info:n ....
.. 1905, 1906, 1913, 1919, 1926, 1932
\_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 .
..... 481, 500, 508
\c_pdf_backend_PageResources_-
clist .. 413, 423, 457, 473, 656, 862
\_pdf_backend_PageResources_-
gpsh:n .....
..... 13, 515, 549, 570, 654, 746, 846
\_pdf_backend_PageResources_-
gpsh_aux:n ..... 837, 863
\_pdf_backend_PageResources_-
gput:nnn 422, 438, 449, 485, 501, 509
\_pdf_backend_PageResources_-
obj_gpsh: . 422, 455, 497, 505, 513
\_pdf_backend_Pages_primitive:n
..... 146, 147, 154, 163, 169, 175
\_pdf_backend_pdfmark:n .....
..... 36, 522, 535, 539, 543, 547, 905
\_pdf_backend_record_abspage:n
..... 65, 76, 208, 764, 794
\_pdf_backend_ref_abspage:n ...
..... 71, 77, 211, 767, 797
\g_pdf_backend_resourceid_int .
..... 88, 207, 208, 211, 755, 760,
764, 767, 775, 785, 790, 794, 797, 805
\_pdf_backend_set_regression_-
data: ..... 1819
\_pdf_backend_shipout_bdc:nn ..
..... 13, 515, 560
\_pdf_backend_structure_-
destination:nn .....
.. 1348, 1360, 1366, 1367, 1472, 1541

```


_pdf_backend_structure_-	destination:nnnn	1349, 1361, 1366, 1457, 1511, 1580	
_pdf_backend_structure_-	destination_aux:nnnn	1411, 1459	
_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, 946, 955, 958, 961, 1001, 1028, 1036, 1039, 1042, 1081, 1192, 1199, 1200, 1201, 1202, 1222, 1231, 1234, 1237, 1241, 1246, 1251, 1255, 1279, 1311, 1320, 1321, 1322, 1323	
\l_pdf_backend_tmpb_box	86, 1256, 1298, 1299, 1300, 1304	
\l_pdf_backend_xform_bool	516, 676, 700, 756, 786, 948, 1030, 1123, 1224	
_pdf_backend_xform_if_exist:n	1333, 1339	
_pdf_backend_xform_new:nnnn	939, 940, 1022, 1110, 1207, 1215	
_pdf_backend_xform_ref:n	939, 1014, 1096, 1143, 1185, 1196, 1209, 1325	
\l_pdf_backend_xform_tmpdp_tl	1213, 1249, 1263, 1270	
\l_pdf_backend_xform_tmphd_tl	1214, 1244, 1268	
\l_pdf_backend_xform_tmpwd_tl	1212, 1239, 1269	
_pdf_backend_xform_use:n	939, 1007, 1087, 1190, 1208, 1309	
\g_pdf_tmpa_prop	82, 226, 231, 236	
\l_pdf_tmpa_tl	82, 209, 213, 215, 218, 765, 769, 771, 774, 795, 799, 801, 804, 807	
pdfannot internal commands:			
_pdfannot_backend_link_begin:n	1464	
_pdfannot_backend_link_-	begin:nnnw	1534, 1603, 1814	
_pdfannot_backend_link_begin_-	goto:nnw	1350, 1356, 1362	
_pdfannot_backend_link_begin_-	structure_goto:nnw	1350, 1356, 1362, 1366, 1462, 1532, 1601, 1812	
_pdfannot_backend_link_off:	922	
_pdfannot_backend_link_on:	926	
pdfdict commands:			
\pdfdict_gput:nnn	190, 218, 320, 359, 395, 440, 451, 503, 511, 678, 702, 758, 773, 788, 803	
\pdfdict_gremove:nn	197, 327, 366, 401	
\pdfdict_if_exist:nTF	213, 769, 799	
\pdfdict_item:nn	236, 842, 857	
\pdfdict_new:n	215, 771, 801	
\pdfdict_put:nnn	1862	
\pdfdict_show:n	807	
\pdfdict_use:n	346, 385, 464, 976, 1057	
\pdfextension	935	
\pdfliteral	2	
pdfmanagement commands:			
\pdfmanagement_add:nnn	1822, 1825, 1828, 1829, 1832, 1840, 1848, 1853	
\pdfnames	20	
\pdfomitinfodict	1915	
\pdfpageref	3	
\pdfrunninglinkoff	920, 924	
\pdfrunninglinkon	928	
\pdftrailerid	1850	
pdfxform commands:			
\pdfxform_dp:n	1146, 1201, 1322	
\pdfxform_ht:n	1145, 1200, 1321	
\pdfxform_if_exist:n	1339	
\pdfxform_wd:n	1144, 1199, 1320	
prg commands:			
\prg_new_conditional:Npnn	1333	
\prg_new_eq_conditional:NNn	1339	
\prg_return_false:	1337	
\prg_return_true:	1336	
prop commands:			
\prop_count:N	970, 1051	
\prop_gclear:N	949, 1031, 1225	
\prop_gput:Nnn	231, 488	
\prop_gset_eq:NN	226	
\prop_if_empty:NTF	459, 658, 839, 980, 985, 990, 995, 1060, 1065, 1070, 1075	
\prop_if_exist:NTF	227, 850	
\prop_map_function:NN	236, 855	
\prop_map_inline:Nn	229	
\prop_new:N	83	
property commands:			
\property_record:nn	68	
\property_ref:nn	73	
\ProvidesExplFile	1	
R			
\relax	132, 1854	

S

scan commands:

\scan_stop: 1011,
1093, 1488, 1508, 1518, 1529, 1557,
1577, 1587, 1598, 1719, 1736, 1745,
1753, 1776, 1793, 1802, 1810, 1849,
1878, 1896, 1902, 1915, 1928, 1941
\setbox 935
\special 2

str commands:

\str_case:nnTF
1374, 1395, 1476, 1496, 1545, 1565,
1615, 1634, 1707, 1724, 1764, 1781
\str_convert_pdfname:n 99, 489
\str_if_eq:nnTF 1273, 1284
sys commands:
\c_sys_engine_exec_str 1864
\sys_gset_rand_seed:n 1821
\c_sys_timestamp_str 1828, 1829, 1862

T

T_EX and L^AT_EX 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, 848
\tex_immediate:D 963, 1044, 1870, 1879
\tex_latelua:D . 249, 285, 302, 684, 708
\tex_luaescapestring:D 244
\tex_pdfcompresslevel:D 1878
\tex_pdfdest:D 1474, 1491,
1513, 1521, 1705, 1720, 1740, 1746
\tex_pdfextension:D
... 46, 56, 880, 1543, 1560, 1582,
1590, 1762, 1777, 1797, 1803, 1870
\tex_pdflastxform:D 1004, 1084

\tex_pdfliteral:D 49, 59
\tex_pdfnames:D 873
\tex_pdfobj:D 1879
\tex_pdfomitcharset:D 1896
\tex_pdfpageattr:D 183
\tex_pdfpageresources:D 848
\tex_pdfpagesattr:D 149
\tex_pdfrefxform:D 1009, 1089
\tex_pdfsuppressptexinfo:D ... 1849
\tex_pdftexrevision:D 1470, 1701, 1911
\tex_pdftexversion:D 1469, 1700, 1910
\tex_pdfvariable:D
..... 1854, 1855, 1902, 1928, 1941
\tex_pdfxform:D 963, 1044
\tex_special:D 40, 165, 313, 352
\tex_the:D
.. 955, 958, 961, 1036, 1039, 1042,
1128, 1131, 1134, 1231, 1234, 1237
\tex_unexpanded:D 244
\tex_vss:D 1421, 1453, 1659, 1688

text commands:

\text_expand:n 99, 105

tl commands:

\c_space_tl 586, 594, 605, 613, 675,
699, 753, 783, 1144, 1145, 1146, 1265
\tl_const:Nn
.. 953, 956, 959, 1034, 1037, 1040,
1126, 1129, 1132, 1229, 1232, 1235
\tl_gput_right:Nn 110, 117, 124
\tl_if_exist:NTF 130
\tl_new:N .. 84, 1212, 1213, 1214, 1343
\tl_set:Nn
..... 209, 765, 795, 1239, 1244, 1249
\tl_to_str:n
..... 954, 957, 960, 1003, 1010,
1016, 1035, 1038, 1041, 1083, 1091,
1097, 1118, 1127, 1130, 1133, 1187,
1219, 1230, 1233, 1236, 1292, 1316,
1327, 1335, 1494, 1524, 1563, 1593
\tl_use:N 1263, 1268, 1269, 1270

U

\unvbox 935

V

\vbox 935

vbox commands:

\vbox_to_zero:n 1413, 1424, 1651, 1662