

The l3backend-testphase package Additional backend PDF features L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.95z, released 2023-08-29

1 l3backend-testphase Implementation

```
1 <drivers>\ProvidesExplFile
2 <*dvipdfmx>
3   {l3backend-testphase-dvipdfmx.def}{2023-08-29}{}
4   {LaTeX-PDF-management-testphase-bundle-backend-support: dvipdfmx}
5 </dvipdfmx>
6 <*dvips>
7   {l3backend-testphase-dvips.def}{2023-08-29}{}
8   {LaTeX-PDF-management-testphase-bundle-backend-support: dvips}
9 </dvips>
10 <*dvisvgm>
11   {l3backend-testphase-dvisvgm.def}{2023-08-29}{}
12   {LaTeX-PDF-management-testphase-bundle-backend-support: dvisvgm}
13 </dvisvgm>
14 <*luatex>
15   {l3backend-testphase-luatex.def}{2023-08-29}{}
16   {LaTeX-PDF-management-testphase-bundle-backend-support: PDF output (LuaTeX)}
17 </luatex>
18 <*pdftex>
19   {l3backend-testphase-pdftex.def}{2023-08-29}{}
20   {LaTeX-PDF-management-testphase-bundle-backend-support: PDF output (pdfTeX)}
21 </pdftex>
22 <*xdvipdfmx>
23   {l3backend-testphase-xetex.def}{2023-08-29}{}
24   {LaTeX-PDF-management-testphase-bundle-backend-support: XeTeX}
25 </xdvipdfmx>
```

1.1 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`.

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

```
26 <*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.
```

```
27 \bool_if:NT \l__pdfmanagement_delayed_shipout_bool
28 {
29   \cs_new_protected:Npn \_kernel_backend_shipout_literal:e #1
30     { \tex_special:D~shipout { #1} }
31 </drivers>
```

(End of definition for _kernel_backend_shipout_literal:e.)

```
32 <*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).
```

```
33 \cs_new_protected:Npn \_kernel_backend_shipout_literal_pdf:e #1
34 {
35 <*luatex>
36   \tex_pdfextension:D ~ literal ~ shipout ~
37 </luatex>
38 <*pdftex>
39   \tex_pdfliteral:D ~ shipout ~
40 </pdftex>
41   { #1 }
42 }
```

(End of definition for _kernel_backend_shipout_literal_pdf:e.)

```
\_kernel_backend_shipout_literal_page:e Page literals are pretty simple.
```

```
43 \cs_new_protected:Npn \_kernel_backend_shipout_literal_page:e #1
44 {
45 <*luatex>
46   \tex_pdfextension:D ~ literal ~ shipout ~
47 </luatex>
48 <*pdftex>
49   \tex_pdfliteral:D ~ shipout ~
50 </pdftex>
51   page { #1 }
52 }
53 </luatex | pdftex>
54 <drivers> }
```

(End of definition for _kernel_backend_shipout_literal_page:e.)

1.2 Crossreferences

This uses the temporary l3ref-tmp.sty. It will be replaced by kernel code later. It is only needed to get a reference for the absolute page counter. This uses the counter from the new lthooks/lthshipout package.

```
55 <@=pdf>
56 <*drivers>
```

```

57 \RequirePackage{l3ref-tmp}
58 \cs_generate_variant:Nn \ref_label:nn {en}
59 \cs_generate_variant:Nn \ref_value:nn {en}
60 \cs_new_protected:Npn \__pdf_backend_ref_label:nn #1 #2
61 {
62   \@bsphack
63   \ref_label:nn{#1}{abspage}
64   \@esphack
65 }
66 \cs_new:Npn \__pdf_backend_ref_value:nn #1 #2
67 {
68   \ref_value:nn{#1}{#2}
69 }
70 \cs_generate_variant:Nn \__pdf_backend_ref_label:nn {en}
71 \cs_generate_variant:Nn \__pdf_backend_ref_value:nn {en}
72 </drivers>

```

avoid that destinations names are optimized with xelatex/dvipdfmx see <https://tug.org/pipermail/dvipdfmx/201905/000002.html>

```

73 <*dvipdfmx |xdvipdfmx>
74   \__kernel_backend_literal:x { dvipdfmx:config-C~ 0x0010 }
75 </dvipdfmx |xdvipdfmx>

```

```

\g__pdf_tmpa_prop   Some scratch variables
  \l__pdf_tmpa_tl
\l__pdf_backend_tmpa_box
76 <*drivers>
77 \prop_new:N \g__pdf_tmpa_prop
78 \tl_new:N   \l__pdf_tmpa_tl
79 \box_new:N \l__pdf_backend_tmpa_box
80 \box_new:N \l__pdf_backend_tmpb_box
81 </drivers>

```

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

```

\g__pdf_backend_resourceid_int  a counter to create labels for the resources, a counter to number properties in bdc marks,
\g__pdf_backend_name_int        a counter for the \pdfpageref implementation.
\g__pdf_backend_page_int
82 <*drivers>
83 \int_new:N \g__pdf_backend_resourceid_int
84 \int_new:N \g__pdf_backend_name_int
85 \int_new:N \g__pdf_backend_page_int
86 </drivers>

```

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

1.3 luacode

Load the lua code.

```

87 <*luatex>
88   \directlua { require("l3backend-testphase.lua") }
89 </luatex>

```

1.4 Converting unicode strings to a pdfname

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

```
90 <*pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
91 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
92 {
93   / \str_convert_pdfname:e { \text_expand:n { #1 } }
94 }
95 </pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
96 <*dvips>
97 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
98 {
99   ~ ( \text_expand:n { #1 } ) ~ cvn
100 }
101 </dvips>
```

1.5 Hooks

1.5.1 Add the “end run” hooks

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

```
102 <*pdftex | luatex>
103 % put in \@kernel@after@enddocument@afterlastpage
104 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
105 {
106   \g__kernel_pdfmanagement_end_run_code_tl
107 }
108 </pdftex | luatex>
109 <*dvipdfmx | xdvipdfmx>
110 % put in \@kernel@after@shipout@lastpage
111 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
112 {
113   \g__kernel_pdfmanagement_end_run_code_tl
114 }
115 </dvipdfmx | xdvipdfmx>
116 <*dvips>
117 % put in \@kernel@after@shipout@lastpage
118 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
119 {
120   \g__kernel_pdfmanagement_end_run_code_tl
121 }
122 </dvips>
```

1.5.2 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 ...

```
123 <*drivers>
124 \tl_if_exist:NTF \@kernel@after@shipout@background
125 {
126   \g@addto@macro \@kernel@before@shipout@background{\relax}
127   \g@addto@macro \@kernel@after@shipout@background
```

```

128     {
129       \g__kernel_pdfmanagement_thispage_shipout_code_tl
130     }
131   }
132   {
133     \hook_gput_code:nnn{shipout/background}{pdf}
134     {
135       \g__kernel_pdfmanagement_thispage_shipout_code_tl
136     }
137   }
138
139 </drivers>

```

1.6 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.

```

140 <*pdftex>
141 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
142   {
143     \tex_global:D \tex_pdfpagesattr:D { #1 }
144   }
145 </pdftex>
146 <*luatex>
147 %luatex: does it in lua
148 \sys_if_engine_luatex:T
149   {
150     \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
151       {
152         \tex_directlua:D
153           {
154             pdf.setpagesattributes( \_pdf_backend_luastring:n { #1 } )
155           }
156       }
157   }
158 </luatex>
159 <*dvips>
160 \cs_new_protected:Npx \_pdf_backend_Pages_primitive:n #1
161   {
162     \tex_special:D{ps:~[#1~/PAGES~pdfmark] %}
163   }
164 </dvips>
165 <*dviPDFmx | xdviPDFmx>
166 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
167   {
168     \_pdf_backend:n{put~@pages~<<#1>>}
169   }
170 </dviPDFmx | xdviPDFmx>
171 <*dvisvgm>
172 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
173   {}

```

174 \langle /dvisvgm \rangle

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

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

```
\_pdf_backend_Page_primitive:n \_pdf_backend_Page_primitive:n is the primitive command to add something to the
\_pdf_backend_Page_gput:nn /Page dictionary. It works differently for the backends: pdftex and luatex overwrite
\_pdf_backend_Page_gremove:n existing content, dvips and dvipdfmx are additive. luatex sets it in lua. The higher
\_pdf_backend_ThisPage_gput:nn level code has to take this into account. \_pdf_backend_Page_gput:nn stores default
\_pdf_backend_ThisPage_gpush:n 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  $\backslash$ g\_pdf_backend_thispage_shipout_tl.

175 % backend commands
176  $\langle$ *pdftex $\rangle$ 
177 %the primitive
178 \cs_new_protected:Npn \_pdf_backend_Page_primitive:n #1
179 {
180     \tex_global:D \tex_pdfpageattr:D { #1 }
181 }
182 % the command to store default values.
183 % Uses a prop with pdflatex + dvi,
184 % sets a lua table with luatex
185 \cs_new_protected:Npn \_pdf_backend_Page_gput:nn #1 #2 %key,value
186 {
187     \pdfdict_gput:nnn {g\_pdf_Core/Page}{ #1 }{ #2 }
188 }
189 % the command to remove a default value.
190 % Uses a prop with pdflatex + dvi,
191 % changes a lua table with luatex
192 \cs_new_protected:Npn \_pdf_backend_Page_gremove:n #1
193 {
194     \pdfdict_gremove:nn {g\_pdf_Core/Page}{ #1 }
195 }
196 % the command used in the document.
197 % direct call of the primitive special with dvips/dvipdfmx
198 % \latelua: fill a page related table with luatex, merge it with the page
199 % table and push it directly
200 % write to aux and store in prop with pdflatex
201 \cs_new_protected:Npn \_pdf_backend_ThisPage_gput:nn #1 #2
202 {
203     %we need to know the page the resource should be added too.
204     \int_gincr:N\g\_pdf_backend_resourceid_int
205     \_pdf_backend_ref_label:en { l3pdf\int_use:N\g\_pdf_backend_resourceid_int }{abspage}
206     \tl_set:Nx \l\_pdf_tmpa_tl
207     {
208         \_pdf_backend_ref_value:en {l3pdf\int_use:N\g\_pdf_backend_resourceid_int}{abspage}
209     }
210     \pdfdict_if_exist:nF { g\_pdf_Core/backend_Page\l\_pdf_tmpa_tl}
211     {
212         \pdfdict_new:n { g\_pdf_Core/backend_Page\l\_pdf_tmpa_tl}
213     }

```

```

214 %backend_Page has no handler.
215 \pdfdict_gput:nnn {g__pdf_Core/backend_Page\l__pdf_tmpa_tl}{ #1 }{ #2 }
216 }
217 %the code to push the values, used in shipout
218 %merges the two props and then fills the register in pdflatex
219 %merges the two tables and then fills (in lua) in luatex
220 %issues the values stored in the global prop with dvi
221 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
222 {
223   \prop_gset_eq:Nc \g__pdf_tmpa_prop { \__kernel_pdfdict_name:n { g__pdf_Core/Page } }
224   \prop_if_exist:cT { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
225   {
226     \prop_map_inline:cn { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
227     {
228       \prop_gput:Nnn \g__pdf_tmpa_prop { ##1 }{ ##2 }
229     }
230   }
231   \exp_args:Nx \__pdf_backend_Page_primitive:n
232   {
233     \prop_map_function:NN \g__pdf_tmpa_prop \pdfdict_item:ne
234   }
235 }
236 </pdftex>
237 <*luatex>
238 % do we need to use some escaping for the values?????
239 \cs_new:Npn \__pdf_backend_luastring:n #1
240 {
241   "\tex_luaescapestring:D { \tex_unexpanded:D { #1 } }"
242 }
243 %not used, only there for consistency
244 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
245 {
246   \tex_latelua:D
247   {
248     pdf.setpageattributes(\__pdf_backend_luastring:n { #1 })
249   }
250 }
251 % the command to store default values.
252 % Uses a prop with pdflatex + dvi,
253 % sets a lua table with luatex
254 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
255 {
256   \tex_directlua:D
257   {
258     ltx.__pdf.backend_Page_gput
259     (
260       \__pdf_backend_luastring:n { #1 },
261       \__pdf_backend_luastring:n { #2 }
262     )
263   }
264 }
265 % the command to remove a default value.
266 % Uses a prop with pdflatex + dvi,
267 % changes a lua table with luatex

```

```

268 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
269 {
270   \tex_directlua:D
271   {
272     ltx.__pdf.backend_Page_gremove (\__pdf_backend_luastring:n { #1 })
273   }
274 }
275 % the command used in the document.
276 % direct call of the primitive special with dvips/dvipdfmx
277 % \lualua: fill a page related table with luatex, merge it with the page
278 % table and push it directly
279 % write to aux and store in prop with pdflatex
280 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
281 {
282   \tex_latelua:D
283   {
284     ltx.__pdf.backend_ThisPage_gput
285     (
286       tex.count["g_shipout_readonly_int"],
287       \__pdf_backend_luastring:n { #1 },
288       \__pdf_backend_luastring:n { #2 }
289     )
290     ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
291   }
292 }
293 %the code to push the values, used in shipout
294 %merges the two props and then fills the register in pdflatex
295 %merges the two tables (the one is probably still empty) and then fills (in lua) in luatex
296 %issues the values stored in the global prop with dvi
297 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
298 {
299   \tex_latelua:D
300   {
301     ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
302   }
303 }
304
305 </luatex>
306 < *dvipdfmx | xdvipdfmx >
307 %the primitive
308 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
309 {
310   \tex_special:D{pdf:-put~@thispage~<<#1>>}
311 }
312 % the command to store default values.
313 % Uses a prop with pdflatex + dvi,
314 % sets a lua table with luatex
315 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
316 {
317   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
318 }
319 % the command to remove a default value.
320 % Uses a prop with pdflatex + dvi,
321 % changes a lua table with luatex

```



```

322 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
323   {
324     \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
325   }
326   % the command used in the document.
327   % direct call of the primitive special with dvips/dvipdfmx
328   % \latalua: fill a page related table with luatex, merge it with the page
329   % table and push it directly
330   % write to aux and store in prop with pdflatex
331 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
332   {
333     \__pdf_backend_Page_primitive:n { /#1~#2 }
334   }
335   %the code to push the values, used in shipout
336   %merges the two props and then fills the register in pdflatex
337   %merges the two tables (the one is probably still empty)
338   % and then fills (in lua) in luatex
339   %issues the values stored in the global prop with dvi
340 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
341   {
342     \exp_args:Nx \__pdf_backend_Page_primitive:n
343       { \pdfdict_use:n { g__pdf_Core/Page} }
344   }
345   </dvipdfmx|xdvipdfmx>
346   <*dvips>
347 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
348   {
349     \tex_special:D{ps:-[ThisPage]<<#1>>~/PUT~pdfmark} %]
350   }
351   % the command to store default values.
352   % Uses a prop with pdflatex + dvi,
353   % sets a lua table with luatex
354 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
355   {
356     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
357   }
358   % the command to remove a default value.
359   % Uses a prop with pdflatex + dvi,
360   % changes a lua table with luatex
361 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
362   {
363     \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
364   }
365   % the command used in the document.
366   % direct call of the primitive special with dvips/dvipdfmx
367   % \latalua: fill a page related table with luatex, merge it with the page
368   % table and push it directly
369   % write to aux and store in prop with pdflatex
370 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
371   {
372     \__pdf_backend_Page_primitive:n { /#1~#2 }
373   }
374   %the code to push the values, used in shipout
375   %merges the two props and then fills the register in pdflatex

```

```

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

```

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

1.8 “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.

```

407 <*drivers>
408 \clist_const:Nn \c__pdf_backend_PageResources_clist
409 {
410   ExtGState,
411   ColorSpace,
412   Pattern,
413   Shading,
414 }
415 </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:`

```

416 <*pdfTeX | luatex>
417 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
418   {
419     \pdf_object_new:n {__pdf/Page/Resources/#1}
420     \cs_if_exist:NT \tex_directlua:D
421       {
422         \tex_directlua:D
423           {
424             ltx.__pdf.object["__pdf/Page/Resources/#1"]
425             =
426             "\_pdf_backend_object_ref:n{__pdf/Page/Resources/#1}"
427           }
428       }
429   }
430 </pdfTeX | luatex>

```

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

```

431 <*luatex>
432 \cs_new_protected:Npn \_pdf_backend_PageResources_gput:nnn #1 #2 #3
433   {
434     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
435     \tex_latelua:D{ltx.__pdf.Page.Resources.#1=true}
436     \tex_latelua:D
437       {
438         ltx.pdf.Page_Resources_gpush(tex.count["g_shipout_readonly_int"])
439       }
440   }
441 </luatex>
442 <*pdfTeX>
443 \cs_new_protected:Npn \_pdf_backend_PageResources_gput:nnn #1 #2 #3
444   {
445     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
446   }
447 </pdfTeX>

```

code for end of document code

```

448 <*pdfTeX | luatex>
449 \cs_new_protected:Npn \_pdf_backend_PageResources_obj_gpush:
450   {
451     \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
452       {
453         \prop_if_empty:cF
454           { \_kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1 } }
455           {
456             \pdf_object_write:nnx
457               { __pdf/Page/Resources/##1 } { dict }

```

```

458         { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1 }
459       }
460     }
461   }
462 </pdfTeX | luatex>

```

xdvipdfmx doesn't work correctly with object names ... <https://tug.org/pipermail/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.

```

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

```

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

```

dvipsvgm unneeded, or no-op

```

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

```

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

1.8.1 Page resources /Properties + BDC operators

```

\__pdf_backend_bdc:nn \__pdf_backend_bdc:nn, \__pdf_backend_shipout_bdc:ee, \__pdf_backend_bdcobject:nn,
\__pdf_backend_shipout_bdc:ee \__pdf_backend_bdcobject:n, \__pdf_backend_bmc:n and \__pdf_backend_emc: are
\__pdf_backend_bdcobject:nn 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 re-
\__pdf_backend_bmc:n sources for the current page.
\__pdf_backend_emc:
\__pdf_backend_PageResources_gpush:n
510 % pdftex and luatex (and perhaps dvips ...) need to know if there are in a
511 % xform stream ...
512 <*drivers>
513 \bool_new:N \l__pdf_backend_xform_bool
514 </drivers>
515 <*dvips>
516 % dvips is easy: create an object, and reference it in the bdc
517 % ghostscript will then automatically replace it by a name
518 % and add the name to the /Properties dict
519 % special variant von accsupp
520 % https://chat.stackexchange.com/transcript/message/50831812#50831812
521 %
522 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
523   {
524     \__pdf_backend_pdfmark:n{/#1~<<#2>>~/BDC}
525   }
526
527 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
528   {
529     \cs_new_protected:Npn \__pdf_backend_bdc_shipout:ee #1 #2 % #1 eg. Span, #2: dict_content
530       {
531         \__kernel_backend_shipout_literal:e
532         {ps: SDict ~ begin ~ mark /#1~<<#2>>~/BDC ~ pdfmark ~ end }
533       }
534     }
535
536 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
537   {
538     \__pdf_backend_pdfmark:x{/#1~\__pdf_backend_object_ref:n{#2}~/BDC}
539   }
540 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
541   {
542     \__pdf_backend_pdfmark:x{/#1~\__pdf_backend_object_last:~/BDC}
543   }
544 \cs_set_protected:Npn \__pdf_backend_emc:

```

```

545 {
546   \_pdf_backend_pdfmark:n{/EMC} %
547 }
548 \cs_set_protected:Npn \_pdf_backend_bmc:n #1
549 {
550   \_pdf_backend_pdfmark:n{/#1~/BMC} %
551 }
552 \cs_new_protected:Npn \_pdf_backend_PageResources_gpush:n #1 {}
553
554 </dvips>
555 <*dvisvgm>
556 % dvisvgm should do nothing
557 %
558 \cs_set_protected:Npn \_pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
559 {}
560 \bool_if:NT\l_pdfmanagement_delayed_shipout_bool
561 {
562   \cs_set_protected:Npn \_pdf_backend_shipout_bdc:ee #1 #2 % #1 eg. Span, #2: dict_content
563   {}
564 }
565 \cs_set_protected:Npn \_pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
566 {}
567 \cs_set_protected:Npn \_pdf_backend_bdcobject:n #1 % #1 eg. Span,
568 {}
569 \cs_set_protected:Npn \_pdf_backend_emc:
570 {}
571 \cs_set_protected:Npn \_pdf_backend_bmc:n #1
572 {}
573 \cs_new_protected:Npn \_pdf_backend_PageResources_gpush:n #1 {}
574
575 </dvisvgm>
576
577 % xetex has to create the entries in the /Properties manually
578 % (like the other backends)
579 % use pdfbase special
580 % https://chat.stackexchange.com/transcript/message/50832016#50832016
581 % the property is added to xform resources automatically,
582 % no need to worry about it.
583 <*dvipdfmx | xdvipdfmx>
584 \cs_set_protected:Npn \_pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
585 {
586   \int_gincr:N \g_pdf_backend_name_int
587   \_kernel_backend_literal:x
588   {
589     pdf:code~/#1/l3pdf\int_use:N\g_pdf_backend_name_int\c_space_tl BDC
590   }
591   \_kernel_backend_literal:x
592   {
593     pdf:put~@resources~
594     <<
595       /Properties~
596       <<
597         /l3pdf\int_use:N\g_pdf_backend_name_int\c_space_tl
598         \_pdf_backend_object_ref:n { #2 }

```

```

599         >>
600     >>
601     }
602 }
603 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span
604 {
605     \int_gincr:N \g__pdf_backend_name_int
606     \__kernel_backend_literal:x
607     {
608         pdf:code~/\exp_not:n{#1}/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
609     }
610     \__kernel_backend_literal:x
611     {
612         pdf:put~@resources~
613         <<
614             /Properties~
615             <<
616                 /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
617                 \__pdf_backend_object_last:
618             >>
619         >>
620     }
621 }
622 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
623 {
624     \__kernel_backend_literal:n {pdf:code~/#1~BMC} %pdfbase
625 }
626
627 %this require management
628 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
629 {
630     \pdf_object_unnamed_write:nn { dict }{ #2 }
631     \__pdf_backend_bdcobject:n { #1 }
632 }
633
634 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
635 {
636     \__kernel_backend_literal:n {pdf:code~/#1~<<#2>>~BDC }
637 }
638
639 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
640 {
641     \bool_if:NTF \g__pdfmanagement_active_bool
642     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
643     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
644     \__pdf_backend_bdc:nn {#1}{#2}
645 }
646
647 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
648 {
649     \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
650     {
651         \__kernel_backend_shipout_literal:e {pdf:code~/#1~<<#2>>~BDC }
652     }

```

```

653 \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
654 }
655 \cs_set_protected:Npn \__pdf_backend_emc:
656 {
657   \__kernel_backend_literal:n {pdf:code~EMC} %pdfbase
658 }
659 % properties are handled automatically, but the other resources should be added
660 % at shipout
661 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
662 {
663   \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
664   {
665     \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
666     {
667       \__kernel_backend_literal:x
668       {
669         pdf:put~@resources~
670         <</##1~\__pdf_backend_object_ref:n {__pdf/Page/Resources/##1}>>
671       }
672     }
673   }
674 }
675 </dviPDFmx | xdvipdfmx>
676 % luatex + pdftex
677 <*luatex>
678 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
679 {
680   \int_gincr:N \g__pdf_backend_name_int
681   \exp_args:Nx\__kernel_backend_literal_page:n
682   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
683   \bool_if:NTF \l__pdf_backend_xform_bool
684   {
685     \exp_args:Nnx\pdfdict_gput:nnn
686     { g__pdf_Core/Xform/Resources/Properties }
687     { l3pdf\int_use:N\g__pdf_backend_name_int }
688     { \__pdf_backend_object_ref:n { #2 } }
689   }
690   {
691     \exp_args:Nx \tex_latelua:D
692     {
693       ltx.pdf.Page_Resources_Properties_gput
694       (
695         tex.count["g_shipout_readonly_int"],
696         "l3pdf\int_use:N\g__pdf_backend_name_int",
697         "\__pdf_backend_object_ref:n { #2 }"
698       )
699     }
700   }
701 }
702 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
703 {
704   \int_gincr:N \g__pdf_backend_name_int
705   \exp_args:Nx\__kernel_backend_literal_page:n
706   { /\exp_not:n{##1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }

```



```

707 \bool_if:NTF \l__pdf_backend_xform_bool
708 {
709   \exp_args:Nnx\pdfdict_gput:nnn %no handler needed
710   { g__pdf_Core/Xform/Resources/Properties }
711   { l3pdf\int_use:N\g__pdf_backend_name_int }
712   { \__pdf_backend_object_last: }
713 }
714 {
715   \exp_args:Nx \tex_latelua:D
716   {
717     ltx.pdf.Page_Resources_Properties_gput
718     (
719       tex.count["g_shipout_readonly_int"],
720       "l3pdf\int_use:N\g__pdf_backend_name_int",
721       "\__pdf_backend_object_last:"
722     )
723   }
724 }
725 }
726 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
727 {
728   \__kernel_backend_literal_page:n { /#1~BMC }
729 }
730 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
731 {
732   \pdf_object_unnamed_write:nn { dict } { #2 }
733   \__pdf_backend_bdcobject:n { #1 }
734 }
735 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
736 {
737   \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
738 }
739
740 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
741 {
742   \bool_if:NTF \g__pdfmanagement_active_bool
743   {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
744   {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
745   \__pdf_backend_bdc:nn {#1}{#2}
746 }
747
748 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
749 {
750   \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
751   {
752     \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
753   }
754   \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
755 }
756
757 \cs_set_protected:Npn \__pdf_backend_emc:
758 {
759   \__kernel_backend_literal_page:n { EMC }
760 }

```

```

761
762 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
763 </luatex>
764 <*pdfTeX>
765 % pdfLaTeX is the most complicated as it has to go through the aux ...
766 % the push command is extended to take other resources too
767 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
768 {
769   \int_gincr:N \g__pdf_backend_name_int
770   \exp_args:Nx\__kernel_backend_literal_page:n
771     { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
772   % code to set the property ....
773   \int_gincr:N\g__pdf_backend_resourceid_int
774   \bool_if:NTF \l__pdf_backend_xform_bool
775     {
776       \exp_args:Nxxx\pdfdict_gput:nnn %no handler needed
777         { g__pdf_Core/Xform/Resources/Properties }
778         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
779         { \__pdf_backend_object_ref:n { #2 } }
780     }
781   {
782     \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
783     \tl_set:Nx \l__pdf_tmpa_tl
784       {
785         \__pdf_backend_ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
786       }
787     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
788       {
789         \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
790       }
791     \exp_args:Nxxx\pdfdict_gput:nnn
792       { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
793       { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
794       { \__pdf_backend_object_ref:n{#2} }
795   }
796 }
797 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
798 {
799   \int_gincr:N \g__pdf_backend_name_int
800   \exp_args:Nx\__kernel_backend_literal_page:n
801     { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
802   % code to set the property ....
803   \int_gincr:N\g__pdf_backend_resourceid_int
804   \bool_if:NTF \l__pdf_backend_xform_bool
805     {
806       \exp_args:Nxxx\pdfdict_gput:nnn
807         { g__pdf_Core/Xform/Resources/Properties }
808         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
809         { \__pdf_backend_object_last: }
810     }
811   {
812     \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
813     \tl_set:Nx \l__pdf_tmpa_tl
814       {

```

```

815         \_pdf_backend_ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspa
816     }
817     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
818     {
819         \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
820     }
821     \exp_args:Nnxx\pdfdict_gput:nnn
822     { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
823     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
824     { \_pdf_backend_object_last: }
825     %\pdfdict_show:n { g_backend_Page\l__pdf_tmpa_tl/Resources/Properties }
826 }
827 }
828 \cs_set_protected:Npn \_pdf_backend_bmc:n #1
829 {
830     \_kernel_backend_literal_page:n { /#1-BMC }
831 }
832 \cs_set_protected:Npn \_pdf_backend_bdc_contobj:nn #1 #2
833 {
834     \pdf_object_unnamed_write:nn { dict } { #2 }
835     \_pdf_backend_bdcobject:n { #1 }
836 }
837 \cs_set_protected:Npn \_pdf_backend_bdc_contstream:nn #1 #2
838 {
839     \_kernel_backend_literal_page:n { /#1-<<#2>>-BDC }
840 }
841
842 \cs_set_protected:Npn \_pdf_backend_bdc:nn #1 #2
843 {
844     \bool_if:NTF \g__pdfmanagement_active_bool
845     {\cs_gset_eq:NN \_pdf_backend_bdc:nn \_pdf_backend_bdc_contobj:nn}
846     {\cs_gset_eq:NN \_pdf_backend_bdc:nn \_pdf_backend_bdc_contstream:nn}
847     \_pdf_backend_bdc:nn {#1}{#2}
848 }
849 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
850 {
851     \cs_set_protected:Npn \_pdf_backend_bdc_shipout_contstream:ee #1 #2
852     {
853         \_kernel_backend_shipout_literal_page:e { /#1-<<#2>>-BDC }
854     }
855     \cs_set_eq:NN \_pdf_backend_bdc_shipout:ee \_pdf_backend_bdc_shipout_contstream:ee
856 }
857
858 \cs_set_protected:Npn \_pdf_backend_emc:
859 {
860     \_kernel_backend_literal_page:n { EMC }
861 }
862
863 \cs_new:Npn \_pdf_backend_PageResources_gpush_aux:n #1 %#1 ExtGState etc
864 {
865     \prop_if_empty:cF
866     { \_kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
867     {
868         \pdfdict_item:ne { #1 }{\pdf_object_ref:n {\_pdf/Page/Resources/#1}}

```

```

869     }
870   }
871
872   \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
873   {
874     \exp_args:NNx \tex_global:D \tex_pdfpageresources:D
875     {
876       \prop_if_exist:cT
877       { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
878       {
879         /Properties~
880         <<
881         \prop_map_function:cN
882         { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Property
883         \pdfdict_item:ne
884         >>
885       }
886       %% add ExtGState etc
887       \clist_map_function:NN
888       \c__pdf_backend_PageResources_clist
889       \__pdf_backend_PageResources_gpush_aux:n
890     }
891   }
892
893   </pdfutex>

```

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

1.9 “Catalog” & subdirectories (pdfcatalog)

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

1.9.1 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.

```

894   % pdflatex
895   <*pdfutex>
896   \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 % #1 name of name tree, #2 array co
897   {
898     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
899     \tex_pdfnames:D {/#1~\pdf_object_ref_last:}
900   }
901   </pdfutex>
902   <*luatex>
903   \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 % #1 name of name tree, #2 array co
904   {
905     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
906     \tex_pdfextension:D~names~ {/#1~\pdf_object_ref_last:}
907   }
908   </luatex>

```

```

909 <*dvipdfmx | xdvipdfmx>
910 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
911 {
912   \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
913   \__pdf_backend:x {put~@names~<</#1~\pdf_object_ref_last: >>}
914 }
915 </dvipdfmx | xdvipdfmx>
916
917 %dvips: noop
918 <*dvips>
919 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
920 </dvips>
921 %dvisvgm: noop
922 <*dvisvgm>
923 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
924 </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.

```

925 <*pdftex | luatex | dvipdfmx | xdvipdfmx>
926 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2 {}
927 </pdftex | luatex | dvipdfmx | xdvipdfmx>
928 <*dvips>
929 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
930 {
931   \__pdf_backend_pdfmark:x
932   {
933     /Name~#1~
934     /FS~#2~
935     /EMBED
936   }
937 }
938 </dvips>
939 <*dvisvgm>
940 %no op. Or is there any sensible use for it?
941 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
942 {}
943
944 </dvisvgm>

```

(End of definition for __pdf_backend_NamesEmbeddedFiles_add:nn.)

1.9.2 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. We add here backend support for this.

```

945 <*drivers>
946 \cs_new_protected:Npn \__pdf_backend_link_off: {}
947 \cs_new_protected:Npn \__pdf_backend_link_on: {}
948 </drivers>
949 <*pdftex>

```

```

950 \cs_if_exist:NT \pdfrunninglinkoff
951 {
952   \cs_set_protected:Npn \__pdf_backend_link_off:
953     {
954       \pdfrunninglinkoff
955     }
956   \cs_set_protected:Npn \__pdf_backend_link_on:
957     {
958       \pdfrunninglinkon
959     }
960 }
961 </pdftex>
962 <*luatex>
963 \int_compare:nNnT {\tex_luatexversion:D } > {112}
964 {
965   \cs_set_protected:Npn \__pdf_backend_link_off:
966     {
967       \pdfextension linkstate 1
968     }
969   \cs_set_protected:Npn \__pdf_backend_link_on:
970     {
971       \pdfextension linkstate 0
972     }
973 }
974 </luatex>
975 <*dviPDFmx | xdvipdfmx>
976 \cs_set_protected:Npn \__pdf_backend_link_off:
977   {
978     \__pdf_backend:n { noline }
979   }
980 \cs_set_protected:Npn \__pdf_backend_link_on:
981   {
982     \__pdf_backend:n { link }
983   }
984 </dviPDFmx | xdvipdfmx>

```

1.9.3 Form XObject / 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 985 <*pdftex>
\__pdf_backend_xform_ref:n 986 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
                              987 % #1 name
                              988 % #2 attributes
                              989 % #3 resources
                              990 % #4 content, not necessarily a box!
                              991 {
                              992   \hbox_set:Nn \l__pdf_backend_tmpa_box
                              993     {
                              994       \bool_set_true:N \l__pdf_backend_xform_bool

```

```

995     \prop_gclear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
996     #4
997   }
998   %store the dimensions
999   \tl_const:cx
1000   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1001   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1002   \tl_const:cx
1003   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1004   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1005   \tl_const:cx
1006   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1007   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1008   %% do we need to test if #2 and #3 are empty??
1009   \tex_immediate:D \tex_pdfxform:D
1010   ~ attr ~ { #2 }
1011   %% which other resources should be default? Is an argument actually needed?
1012   ~ resources ~
1013   {
1014     #3
1015     \int_compare:nNnT
1016     { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1017     >
1018     { 0 }
1019     {
1020       /Properties~
1021       <<
1022         \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1023       >>
1024     }
1025
1026     \prop_if_empty:cF
1027     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1028     {
1029       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1030     }
1031
1032     \prop_if_empty:cF
1033     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1034     {
1035       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1036     }
1037
1038     \prop_if_empty:cF
1039     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1040     {
1041       /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1042     }
1043
1044     \prop_if_empty:cF
1045     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1046     {
1047       /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1048     }
1049   }
1050   \l__pdf_backend_tmpa_box
1051   \int_const:cn

```

```

1049     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1050     { \tex_pdflastxform:D }
1051   }
1052
1053 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1054 {
1055   \tex_pdfrefxform:D
1056   \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1057   \scan_stop:
1058 }
1059
1060 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1061 {
1062   \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
1063 }
1064 </pdfTeX>
1065 <*luatex>
1066 %luatex
1067 %nearly identical but not completely ...
1068 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1069 % #1 name
1070 % #2 attributes
1071 % #3 resources
1072 % #4 content, not necessarily a box!
1073 {
1074   \hbox_set:Nn \l__pdf_backend_tmpa_box
1075   {
1076     \bool_set_true:N \l__pdf_backend_xform_bool
1077     \prop_gclear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }
1078     #4
1079   }
1080   \tl_const:cx
1081   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1082   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1083   \tl_const:cx
1084   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1085   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1086   \tl_const:cx
1087   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1088   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1089   %% do we need to test if #2 and #3 are empty??
1090   \tex_immediate:D \tex_pdfxform:D
1091   ~ attr ~ { #2 }
1092   %% which resources should be default? Is an argument actually needed?
1093   ~ resources ~
1094   {
1095     #3
1096     \int_compare:nNnT
1097     {\prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties
1098     >
1099     { 0 }
1100     {
1101       /Properties~
1102     <<

```



```

1103         \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1104         >>
1105     }
1106     \prop_if_empty:cF
1107     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1108     {
1109         /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1110     }
1111     \prop_if_empty:cF
1112     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1113     {
1114         /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1115     }
1116     \prop_if_empty:cF
1117     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1118     {
1119         /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1120     }
1121     \prop_if_empty:cF
1122     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1123     {
1124         /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1125     }
1126 }
1127 \l__pdf_backend_tmpa_box
1128 \int_const:cn
1129 { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1130 { \tex_pdflastxform:D }
1131 }
1132
1133 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 %protected as with xelatex
1134 {
1135     \tex_pdfrefxform:D \int_use:c
1136     {
1137         c__pdf_backend_xform_ \tl_to_str:n {#1} _int
1138     }
1139     \scan_stop:
1140 }
1141
1142 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1143 { \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
1144
1145 </luatex>
1146 <*dvi.pdfmx | xdvipdfmx>
1147 % xetex
1148 % it needs a bit testing if it really works to set the box to 0 before the special ...
1149 % does it disturb viewing the xobject?
1150 % what happens with the resources (bdc)? (should work as they are specials too)
1151 % xetex requires that the special is in horizontal mode. This means it affects
1152 % typesetting. But we can no delay the whole form code to shipout
1153 % as the object reference and the size is often wanted on the current page.
1154 % so we need to allocate a box - but probably they won't be thousands xform
1155 % in a document so it shouldn't matter.
1156 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4

```

```

1157 % #1 name
1158 % #2 attributes
1159 % #3 resources
1160 % #4 content, not necessarily a box!
1161 {
1162   \int_gincr:N \g__pdf_backend_object_int
1163   \int_const:cn
1164     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1165     { \g__pdf_backend_object_int }
1166   \box_new:c { g__pdf_backend_xform_#1_box }
1167   \hbox_gset:cn { g__pdf_backend_xform_#1_box }
1168     {
1169       \bool_set_true:N \l__pdf_backend_xform_bool
1170       #4
1171     }
1172   \tl_const:cx
1173     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1174     { \tex_the:D \box_wd:c { g__pdf_backend_xform_#1_box } }
1175   \tl_const:cx
1176     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1177     { \tex_the:D \box_ht:c { g__pdf_backend_xform_#1_box } }
1178   \tl_const:cx
1179     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1180     { \tex_the:D \box_dp:c { g__pdf_backend_xform_#1_box } }
1181   \box_set_dp:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1182   \box_set_ht:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1183   \box_set_wd:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1184   \hook_gput_next_code:nn {shipout/background}
1185   {
1186     \mode_leave_vertical: %needed, the xform disappears without it.
1187     \__pdf_backend:x
1188       {
1189         bXObject ~ \__pdf_backend_xform_ref:n { #1 }
1190         \c_space_tl width ~ \pdfxform_wd:n { #1 }
1191         \c_space_tl height ~ \pdfxform_ht:n { #1 }
1192         \c_space_tl depth ~ \pdfxform_dp:n { #1 }
1193       }
1194     \box_use_drop:c { g__pdf_backend_xform_#1_box }
1195     \__pdf_backend:x {put ~ @resources ~<<#3>> }
1196     \__pdf_backend:x
1197       {
1198         put~ @resources ~
1199         <<
1200           /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1201         >>
1202       }
1203     \__pdf_backend:x
1204       {
1205         put~ @resources ~
1206         <<
1207           /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1208         >>
1209       }
1210     \__pdf_backend:x

```

```

1211     {
1212     put~ @resources ~
1213     <<
1214     /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1215     >>
1216     }
1217     \__pdf_backend:x
1218     {
1219     put~ @resources ~
1220     <<
1221     /ColorSpace~
1222     \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1223     >>
1224     }
1225     \exp_args:Nx
1226     \__pdf_backend:x {exobj ~<<#2>>}
1227   }
1228 }
1229
1230
1231
1232 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1233 {
1234   @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1235 }
1236
1237 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1238 {
1239   \hbox_set:Nn \l__pdf_backend_tmpa_box
1240   {
1241     \__pdf_backend:x
1242     {
1243       uxobj~ \__pdf_backend_xform_ref:n { #1 }
1244     }
1245   }
1246   \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1247   \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1248   \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1249   \box_use_drop:N \l__pdf_backend_tmpa_box
1250 }
1251 </dviptfm x|xdviptfm x>
1252 <*dvisvgm>
1253 % unclear what it should do!!
1254 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}
1255 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
1256 \cs_new:Npn \__pdf_backend_xform_ref:n {}
1257 </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

```

1258 <*dvips>
1259 \tl_new:N \l__pdf_backend_xform_tmpwd_tl

```

```

1260 \tl_new:N \l__pdf_backend_xform_tmpdp_tl
1261 \tl_new:N \l__pdf_backend_xform_tmplt_tl
1262 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 % #1 name, #2 attribute, #4
1263 {
1264   \int_gincr:N \g__pdf_backend_object_int
1265   \int_const:cn
1266   { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1267   { \g__pdf_backend_object_int }
1268
1269   \hbox_set:Nn \l__pdf_backend_tmpa_box
1270   {
1271     \bool_set_true:N \l__pdf_backend_xform_bool
1272     \prop_gc clear:c {\__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
1273     #4
1274   }
1275   %store the dimensions
1276   \tl_const:cx
1277   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1278   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1279   \tl_const:cx
1280   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1281   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1282   \tl_const:cx
1283   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1284   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1285   %store content dimensions in DPI units (Dots) (code from issue 25)
1286   \tl_set:Nx \l__pdf_backend_xform_tmpwd_tl
1287   {
1288     \dim_to_decimal_in_sp:n{ \box_wd:N \l__pdf_backend_tmpa_box }~
1289     65536~div~72.27~div~DVImag~mul~Resolution~mul~
1290   }
1291   \tl_set:Nx \l__pdf_backend_xform_tmplt_tl
1292   {
1293     \dim_to_decimal_in_sp:n{ \box_ht:N \l__pdf_backend_tmpa_box }~
1294     65536~div~72.27~div~DVImag~mul~VResolution~mul~
1295   }
1296   \tl_set:Nx \l__pdf_backend_xform_tmpdp_tl
1297   {
1298     \dim_to_decimal_in_sp:n{ \box_dp:N \l__pdf_backend_tmpa_box }~
1299     65536~div~72.27~div~DVImag~mul~VResolution~mul~
1300   }
1301   % mirror the box
1302   %\box_scale:Nnn \l__pdf_backend_tmpa_box {1} {-1}
1303   \hbox_set:Nn \l__pdf_backend_tmpb_box
1304   {
1305     \__kernel_backend_postscript:x
1306     {
1307       gsave~currentpoint~
1308       initclip~ % restore default clipping path (page device/whole page)
1309       clippath~pathbbox~newpath~pop~pop~
1310       \tl_use:N \l__pdf_backend_xform_tmpdp_tl~add~translate~
1311       mark~
1312       /objdef~{ pdf.obj \int_use:N \g__pdf_backend_object_int } \c_space_tl~
1313       /BBox[

```

```

1314         0~
1315         \tl_use:N\l__pdf_backend_xform_tmpt_tl~
1316         \tl_use:N\l__pdf_backend_xform_tmpwd_tl~
1317         \tl_use:N\l__pdf_backend_xform_tmpdp_tl~
1318         neg
1319     ]
1320     \str_if_eq:eeF{#1}{}
1321     {
1322         product~(Distiller)~search~{pop~pop~pop~#2}{pop}ifelse~
1323     }
1324     /BP~pdfmark~1~-1~scale~neg~exch~neg~exch~translate
1325     }
1326     \box_use_drop:N\l__pdf_backend_tmpa_box
1327     \__kernel_backend_postscript:n
1328     {
1329         mark ~ /EP~pdfmark ~ grestore
1330     }
1331     \str_if_eq:eeF{#1}{}
1332     {
1333         \__kernel_backend_postscript:x
1334         {
1335             product~(Ghostscript)~search~
1336             {
1337                 pop~pop~pop~
1338                 mark~
1339                 { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1340                 ~<<#2>>~/PUT~pdfmark
1341             }{pop}ifelse
1342         }
1343     }
1344     }
1345     \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1346     \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1347     \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1348     \hook_gput_code:nnn {begindocument/end}{pdfxform}
1349     {
1350         \mode_leave_vertical:
1351         \box_use:N\l__pdf_backend_tmpb_box
1352     }
1353 }
1354
1355
1356 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1357 {
1358     \hbox_set:Nn \l__pdf_backend_tmpa_box
1359     {
1360         \__kernel_backend_postscript:x
1361         {
1362             gsave~currentpoint~translate~1~-1~scale~
1363             mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }~
1364             /SP~pdfmark ~ grestore
1365         }
1366     }
1367     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }

```

```

1368 \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1369 \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1370 \box_use_drop:N \l__pdf_backend_tmpa_box
1371 }
1372 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1373 {
1374   { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1375 }
1376
1377 </dvips>
1378 <*drivers>
1379 %% all
1380 \prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
1381 {
1382   \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1383     { \prg_return_true: }
1384     { \prg_return_false:}
1385 }
1386 \prg_new_eq_conditional:NNn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
1387 { TF , T , F , p }
1388 </drivers>

```

(End of definition for __pdf_backend_xform_new:n, __pdf_backend_xform_use:n, and __pdf_backend_xform_ref:n.)

1.10 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 TeXlive 2022 (earlier in miktex) both engine will know new keywords which allow to create structure destination easily.

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

`\l_pdf_current_structure_destination_tl` This command holds the name of the structure object to use in the next command 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
```

```

1389 <*drivers>
1390 \tl_new:N \l_pdf_current_structure_destination_tl
1391 </drivers>

```

(End of definition for `\l_pdf_current_structure_destination_tl`. This function is documented on page ??.)

We will define alternatives for three backend commands:

```

\__pdf_backend_destination:nn      -> \__pdf_backend_structure_destination:nn
\__pdf_backend_destination:nmnn -> \__pdf_backend_structure_destination:nmnn
\__pdf_backend_link_begin_goto:nmw -> \__pdf_backend_link_begin_structure_goto:nmw

```

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:`

```

1392 <*drivers>
1393 \cs_new_protected:Npn \pdf_activate_structure_destination:
1394 {
1395   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_structure_destination:nn
1396   \cs_gset_eq:NN \__pdf_backend_destination:nmnn \__pdf_backend_structure_destination:nmnn
1397   \cs_gset_eq:NN \__pdf_backend_link_begin_goto:nmw \__pdf_backend_link_begin_structure_goto:nmw
1398 }
1399 </drivers>

```

(End of definition for `\pdf_activate_structure_destination:`. This function is documented on page ??.)

Now the driver dependant 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.

```

1400 <*drivers>
1401 \cs_set_eq:NN \__pdf_backend_structure_destination:nn \__pdf_backend_destination:nn
1402 \cs_set_eq:NN \__pdf_backend_structure_destination:nmnn \__pdf_backend_destination:nmnn
1403 \cs_set_eq:NN \__pdf_backend_link_begin_structure_goto:nmw \__pdf_backend_link_begin_goto:nmw
1404 </drivers>

```

`__pdf_backend_structure_destination:nn`

This command is the backend command to create a destination. It should in parallel 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:

```

1405 <*xdvipdfmx|dvipdfmx>
1406 \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1407 {
1408   \__pdf_backend:x
1409   {
1410     dest ~ ( \exp_not:n {#1} )
1411     [
1412       @thispage
1413       \str_case:nmF {#2}
1414       {
1415         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1416         { fit } { /Fit }
1417         { fitb } { /FitB }
1418         { fitbh } { /FitBH }

```

```

1419         { fitbv } { /FitBV ~ @xpos }
1420         { fith } { /FitH ~ @ypos }
1421         { fitv } { /FitV ~ @xpos }
1422         { fitr } { /Fit }
1423     }
1424     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1425 ]
1426 }

```

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.

```

1427 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1428 {
1429     \__pdf_backend:x
1430     {
1431         obj ~ @pdf.SDest.\exp_not:n{#1}
1432         [
1433             \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1434             \str_case:nnF {#2}
1435             {
1436                 { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1437                 { fit } { /Fit }
1438                 { fitb } { /FitB }
1439                 { fitbh } { /FitBH }
1440                 { fitbv } { /FitBV ~ @xpos }
1441                 { fith } { /FitH ~ @ypos }
1442                 { fitv } { /FitV ~ @xpos }
1443                 { fitr } { /Fit }
1444             }
1445             { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1446         ]
1447     }
1448 }
1449 }

```

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

```

1450 \cs_new_protected:Npn \__pdf_backend_structure_destination_aux:nnnn #1#2#3#4
1451 {
1452     \vbox_to_zero:n
1453     {
1454         \__kernel_kern:n {#4}
1455         \hbox:n
1456         {
1457             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1458             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1459         }
1460         \tex_vss:D
1461     }
1462     \__kernel_kern:n {#1}
1463     \vbox_to_zero:n
1464     {
1465         \__kernel_kern:n { -#3 }
1466         \hbox:n

```



```

1467     {
1468         \_pdf_backend:n
1469         {
1470             dest ~ (#2)
1471             [
1472                 @thispage
1473                 /FitR ~
1474                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1475                 @xpos ~ @ypos
1476             ]
1477         }

```

Here we add the structure destination to the same box

```

1478         \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1479         {
1480             \_pdf_backend:x
1481             {
1482                 obj ~ @pdf.SDest.\exp_not:n{#2}
1483                 [
1484                     \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_
1485                     /FitR ~
1486                     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1487                     @xpos ~ @ypos
1488                 ]
1489             }
1490         }
1491     }
1492     \tex_vss:D
1493 }
1494 \_kernel_kern:n { -#1 }
1495 }

```

And now we redefine the destination command:

```

1496 \cs_set_protected:Npn \_pdf_backend_structure_destination:nmmn #1#2#3#4
1497 {
1498     \exp_args:Ne \_pdf_backend_structure_destination_aux:nmmn
1499     { \dim_eval:n {#2} } {#1} {#3} {#4}
1500 }

```

At last the goto link.

```

1501 \cs_set_protected:Npn \_pdf_backend_link_begin_structure_goto:nmw #1#2
1502 {
1503     \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD~@pdf.SDest.
1504 }
1505 </xdvipdfmx | dvipdfmx>

```

(End of definition for _pdf_backend_structure_destination:mn.)

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

```

1506 <*pdftex>
1507 \bool_lazy_and:nnT
1508 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1509 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
1510 {
1511     \cs_set_protected:Npn \_pdf_backend_structure_destination:nn #1#2
1512     {

```

```

1513     \tex_pdfdest:D
1514     name {#1}
1515     \str_case:nnF {#2}
1516     {
1517         { xyz } { xyz }
1518         { fit } { fit }
1519         { fitb } { fitb }
1520         { fitbh } { fitbh }
1521         { fitbv } { fitbv }
1522         { fith } { fith }
1523         { fitv } { fitv }
1524         { fitr } { fitr }
1525     }
1526     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1527     \scan_stop:
1528     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1529     {
1530         \tex_pdfdest:D
1531         struct~
1532         \int_use:c
1533         { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1534         name {#1}
1535         \str_case:nnF {#2}
1536         {
1537             { xyz } { xyz }
1538             { fit } { fit }
1539             { fitb } { fitb }
1540             { fitbh } { fitbh }
1541             { fitbv } { fitbv }
1542             { fith } { fith }
1543             { fitv } { fitv }
1544             { fitr } { fitr }
1545         }
1546         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1547         \scan_stop:
1548     }
1549 }
1550 \cs_set_protected:Npn \__pdf_backend_destination:nmmn #1#2#3#4
1551 {
1552     \tex_pdfdest:D
1553     name {#1}
1554     fitr ~
1555     width \dim_eval:n {#2} ~
1556     height \dim_eval:n {#3} ~
1557     depth \dim_eval:n {#4} \scan_stop:
1558     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1559     {
1560         \tex_pdfdest:D
1561         struct~
1562         \int_use:c
1563         { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1564         name {#1}
1565         fitr ~
1566         width \dim_eval:n {#2} ~

```

```

1567         height \dim_eval:n {#3} ~
1568         depth \dim_eval:n {#4} \scan_stop:
1569     }
1570 }
1571 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1572 {
1573     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1574 }
1575 }
1576 </pdfTeX>

```

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

```

1577 <*luatex>
1578 \int_compare:nNnT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1579 {
1580     \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1581     {
1582         \tex_pdfextension:D dest
1583         name {#1}
1584         \str_case:nnF {#2}
1585         {
1586             { xyz } { xyz }
1587             { fit } { fit }
1588             { fitb } { fitb }
1589             { fitbh } { fitbh }
1590             { fitbv } { fitbv }
1591             { fith } { fith }
1592             { fitv } { fitv }
1593             { fitr } { fitr }
1594         }
1595         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1596     \scan_stop:
1597     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1598     {
1599         \tex_pdfextension:D dest
1600         struct~
1601         \int_use:c
1602         { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1603         name {#1}
1604         \str_case:nnF {#2}
1605         {
1606             { xyz } { xyz }
1607             { fit } { fit }
1608             { fitb } { fitb }
1609             { fitbh } { fitbh }
1610             { fitbv } { fitbv }
1611             { fith } { fith }
1612             { fitv } { fitv }
1613             { fitr } { fitr }
1614         }
1615         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1616     \scan_stop:
1617     }
1618 }
1619 \cs_set_protected:Npn \__pdf_backend_destination:nmnn #1#2#3#4

```

```

1620 {
1621   \tex_pdfextension:D dest
1622   name {#1}
1623   fitr ~
1624   width \dim_eval:n {#2} ~
1625   height \dim_eval:n {#3} ~
1626   depth \dim_eval:n {#4} \scan_stop:
1627   \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1628   {
1629     \tex_pdfextension:D dest
1630     struct~
1631     \int_use:c
1632     { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n { \l_pdf_current_structure_
1633     name {#1}
1634     fitr ~
1635     width \dim_eval:n {#2} ~
1636     height \dim_eval:n {#3} ~
1637     depth \dim_eval:n {#4} \scan_stop:
1638   }
1639 }
1640 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1641 {
1642   \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1643 }
1644 }
1645 </luatex>

```

1.11 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 dependant part. The main command is then in `l3pdfmeta`

```

1646 <*drivers>
1647 \cs_new_protected:Npn \__pdf_backend_set_regression_data:
1648 {
1649   \sys_gset_rand_seed:n{1000}
1650   \pdfmanagement_add:nnn{Info}{Creator}{(TeX)}
1651 </drivers>
1652 <*dvips>
1653 \AddToHook{begindocument}{\pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX+dvips)}}
1654 \__kernel_backend_literal:e{!~<</DocumentUUID~(DocumentUUID)>>~setpagedevice}
1655 \__kernel_backend_literal:e{!~<</InstanceUUID~(InstanceUUID)>>~setpagedevice}
1656 \str_if_exist:NTF\c_sys_timestamp_str
1657 {
1658   \pdfmanagement_add:nnx{Info}{CreationDate}{(\c_sys_timestamp_str)}
1659   \pdfmanagement_add:nnx{Info}{ModDate}{(\c_sys_timestamp_str)}
1660 }
1661 {
1662   \pdfmanagement_add:nnn{Info}{CreationDate}{(D:20010101205959-00'00')}
1663   \pdfmanagement_add:nnn{Info}{ModDate}{(D:20010101205959-00'00')}
1664 }
1665 </dvips>
1666 <*dviPDFmx>

```

```

1667   \pdfmanagement_add:nnn{Info}{Producer}{(dvi)pdfmx}
1668   \__kernel_backend_literal:e
1669     {pdf:trailerid [~
1670       <00112233445566778899aabbccddeeff>~
1671       <00112233445566778899aabbccddeeff>~
1672     ]}
1673 </dvi)pdfmx>
1674 <*(x)dvi)pdfmx>
1675   \pdfmanagement_add:nnn{Info}{Producer}{(x)etex}
1676   \__kernel_backend_literal:e
1677     {pdf:trailerid [~
1678       <00112233445566778899aabbccddeeff>~
1679       <00112233445566778899aabbccddeeff>~
1680     ]}
1681 </*(x)dvi)pdfmx>
1682 <*(pdf)tex>
1683   \pdfmanagement_add:nnn{Info}{Producer}{(pdf)TeX}
1684   \tex_pdfsuppressptexinfo:D 7 \scan_stop:
1685   \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
1686 </*(pdf)tex>
1687 <*(lua)tex>
1688   \pdfmanagement_add:nnn{Info}{Producer}{(Lua)TeX}
1689   \tex_pdfvariable:D suppressoptionalinfo 7\relax
1690   \tex_pdfvariable:D trailerid
1691     {[~
1692       <2350CAD05F8A7AF0AA4058486855344F>~
1693       <2350CAD05F8A7AF0AA4058486855344F>~
1694     ]}
1695 </*(lua)tex>
1696 <*(drivers)>
1697   \str_if_exist:NF\c_sys_timestamp_str
1698   {
1699     \pdfmanagement_add:nnn{Info}{CreationDate}{(D:20010101205959-00'00')}
1700     \pdfmanagement_add:nnn{Info}{ModDate}{(D:20010101205959-00'00')}
1701     \AddToDocumentProperties[document]{creationdate}{D:20010101205959-00'00'}
1702     \AddToDocumentProperties[document]{moddate}{D:20010101205959-00'00'}
1703     \AddToDocumentProperties[hyperref]{pdfmetadate}{D:20010101205959-00'00'}
1704     \AddToDocumentProperties[hyperref]{pdfdate}{D:20010101205959-00'00'}
1705   }
1706   \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-8c19-6237d832f}
1707   \AddToDocumentProperties[hyperref]{pdfproducer}{\c_sys_engine_exec_str-NN.NN.NN}
1708 }
1709 </*(drivers)>

```

1.12 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)dvi)pdfmx` 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`:

```

1710 <*luatex>
1711 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1712 {
1713   \tex_immediate:D \tex_pdfextension:D obj ~uncompressed-
1714   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1715 }
1716 </luatex>
1717 <*pdftex>
1718 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1719 {
1720   \group_begin:
1721   \tex_pdfcompresslevel:D 0 \scan_stop:
1722   \tex_immediate:D \tex_pdfobj:D
1723   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1724   \group_end:
1725 }
1726 </pdftex>
1727 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1728 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1729 {
1730   \pdf_object_unnamed_write:nn {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
1731 }
1732 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>

```

1.13 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.

```

1733 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1734 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 {} %#1 number
1735 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1736 <*pdftex>
1737 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 %#1 number
1738 {
1739   \tex_pdfomitcharset:D = #1 \scan_stop:
1740 }
1741 </pdftex>
1742 <*luatex>
1743 \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 %#1 number
1744 {
1745   \tex_pdfvariable:D omitcharset = #1 \scan_stop:
1746 }
1747 </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.

```

1748 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1749 \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {} %#1 number

```

```

1750 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1751 <*pdftex>
1752 \bool_lazy_and:nnTF
1753 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1754 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {24} }
1755 {
1756   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 %#1 number
1757   {
1758     \pdfomitinfodict = #1 \scan_stop:
1759   }
1760 }
1761 {
1762   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {}%#1 number
1763 }
1764 }
1765 </pdftex>
1766 <*luatex>
1767 \int_compare:nNnTF {\directlua{tex.print(status.list()["development_id"])} } > {7560}
1768 {
1769   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 %#1 number
1770   {
1771     \tex_pdfvariable:D omitinfodict = #1 \scan_stop:
1772   }
1773 }
1774 {
1775   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {} %#1 number
1776 }
1777 </luatex>

```

(End of definition for __pdf_backend_omit_info:n.)

1.14 lua code for lualatex

```

1778 <*lua>
1779 ltx= ltx or {}
1780 ltx.__pdf      = ltx.__pdf or {}
1781 ltx.__pdf.Page = ltx.__pdf.Page or {}
1782 ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
1783 ltx.__pdf.Page.Resources = ltx.__pdf.Page.Resources or {}
1784 ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
1785 ltx.__pdf.Page.Resources.List={"ExtGState","ColorSpace","Pattern","Shading"}
1786 ltx.__pdf.object = ltx.__pdf.object or {}
1787
1788 ltx.pdf= ltx.pdf or {} -- for "public" functions
1789
1790 local __pdf = ltx.__pdf
1791 local pdf = pdf
1792
1793 local function __pdf_backend_Page_gput (name,value)
1794   __pdf.Page.dflt[name]=value
1795 end
1796
1797 local function __pdf_backend_Page_gremove (name)
1798   __pdf.Page.dflt[name]=nil

```

```

1799 end
1800
1801 local function __pdf_backend_Page_gclear ()
1802 __pdf.Page.dflt={}
1803 end
1804
1805 local function __pdf_backend_ThisPage_gput (page,name,value)
1806 __pdf.Page[page] = __pdf.Page[page] or {}
1807 __pdf.Page[page][name]=value
1808 end
1809
1810 local function __pdf_backend_ThisPage_gpush (page)
1811 local token=""
1812 local t = {}
1813 local tkeys= {}
1814 for name,value in pairs(__pdf.Page.dflt) do
1815 t[name]=value
1816 end
1817 if __pdf.Page[page] then
1818 for name,value in pairs(__pdf.Page[page]) do
1819 t[name] = value
1820 end
1821 end
1822 -- sort the table to get reliable test files.
1823 for name,value in pairs(t) do
1824 table.insert(tkeys,name)
1825 end
1826 table.sort(tkeys)
1827 for _,name in ipairs(tkeys) do
1828 token = token .. "/"..name.." "..t[name]
1829 end
1830 return token
1831 end
1832
1833 function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly"]
1834 __pdf_backend_ThisPage_gput (page,name,value)
1835 end
1836
1837 function ltx.__pdf.backend_ThisPage_gpush (page)
1838 pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
1839 end
1840
1841 function ltx.__pdf.backend_Page_gput (name,value)
1842 __pdf_backend_Page_gput (name,value)
1843 end
1844
1845 function ltx.__pdf.backend_Page_gremove (name)
1846 __pdf_backend_Page_gremove (name)
1847 end
1848
1849 function ltx.__pdf.backend_Page_gclear ()
1850 __pdf_backend_Page_gclear ()
1851 end
1852

```



```

1853
1854 local Properties = ltx.__pdf.Page.Resources.Properties
1855 local ResourceList= ltx.__pdf.Page.Resources.List
1856 local function __pdf_backend_PageResources_gpush (page)
1857   local token=""
1858   if Properties[page] then
1859     -- we sort the table, so that the pdf test works
1860     local t = {}
1861     for name,value in pairs (Properties[page]) do
1862       table.insert (t,name)
1863     end
1864     table.sort (t)
1865     for _,name in ipairs(t) do
1866       token = token .. "/"..name.." ".. Properties[page][name]
1867     end
1868     token = "/Properties <<"..token..">>"
1869   end
1870   for i,name in ipairs(ResourceList) do
1871     if ltx.__pdf.Page.Resources[name] then
1872       token = token .. "/"..name.." "..ltx.pdf.object_ref("__pdf/Page/Resources/"..name)
1873     end
1874   end
1875   return token
1876 end
1877
1878 -- the function is public, as I probably need it in tagpdf too ...
1879 function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
1880   Properties[page] = Properties[page] or {}
1881   Properties[page][name]=value
1882   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
1883 end
1884
1885 function ltx.pdf.Page_Resources_gpush(page)
1886   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
1887 end
1888
1889 function ltx.pdf.object_ref (objname)
1890   if ltx.__pdf.object[objname] then
1891     local ref= ltx.__pdf.object[objname]
1892     return ref
1893   else
1894     return "false"
1895   end
1896 end
1897 </lua>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A

`\AddToDocumentProperties`
 .. 1701, 1702, 1703, 1704, 1706, 1707₄₁

<code>\AddToHook</code>	1653	740, 750, 757, 767, 797, 828, 832, 837, 842, 851, 858, 952, 956, 965, 969, 976, 980, 1406, 1496, 1501, 1511, 1550, 1571, 1580, 1619, 1640
B		
bool commands:		
<code>\bool_if:NTF</code>	27, 527, 560, 641, 647, 683, 707, 742, 748, 774, 804, 844, 849	
<code>\bool_lazy_and:nnTF</code> 1507, 1752	
<code>\bool_new:N</code> 513	
<code>\bool_set_true:N</code>	994, 1076, 1169, 1271	
box commands:		
<code>\box_dp:N</code>	1007, 1088, 1180, 1284, 1298	
<code>\box_ht:N</code>	1004, 1085, 1177, 1281, 1293	
<code>\box_new:N</code> 79, 80, 1166	
<code>\box_scale:Nnn</code> 1302	
<code>\box_set_dp:Nn</code>	1181, 1248, 1345, 1369	
<code>\box_set_ht:Nn</code>	1182, 1247, 1346, 1368	
<code>\box_set_wd:Nn</code>	1183, 1246, 1347, 1367	
<code>\box_use:N</code> 1351	
<code>\box_use_drop:N</code>	1194, 1249, 1326, 1370	
<code>\box_wd:N</code>	1001, 1082, 1174, 1278, 1288	
C		
clist commands:		
<code>\clist_const:Nn</code> 408	
<code>\clist_map_function:NN</code> 887	
<code>\clist_map_inline:Nn</code>	417, 451, 467, 663	
cs commands:		
<code>\cs_generate_variant:Nn</code>	58, 59, 70, 71	
<code>\cs_gset_eq:NN</code> 642, 643, 743, 744, 845, 846, 1395, 1396, 1397	
<code>\cs_if_exist:NTF</code> 420, 950	
<code>\cs_new:Npn</code> 66, 91, 97, 239, 863, 1060, 1142, 1232, 1256, 1372	
<code>\cs_new_protected:Npn</code> 29, 33, 43, 60, 141, 150, 166, 172, 178, 185, 192, 201, 221, 244, 254, 268, 280, 297, 308, 315, 322, 331, 340, 347, 354, 361, 370, 379, 387, 390, 396, 401, 404, 432, 443, 449, 475, 479, 492, 495, 496, 500, 503, 504, 508, 529, 552, 573, 661, 762, 872, 896, 903, 910, 919, 923, 926, 929, 941, 946, 947, 986, 1053, 1068, 1133, 1156, 1237, 1254, 1255, 1262, 1356, 1393, 1450, 1647, 1711, 1718, 1728, 1734, 1737, 1743, 1749, 1756, 1762, 1769, 1775	
<code>\cs_new_protected:Npx</code> 160	
<code>\cs_set_eq:NN</code> 653, 754, 855, 1401, 1402, 1403	
<code>\cs_set_protected:Npn</code>	522, 536, 540, 544, 548, 558, 562, 565, 567, 569, 571, 584, 603, 622, 628, 634, 639, 649, 655, 678, 702, 726, 730, 735,	
D		
dim commands:		
<code>\dim_eval:n</code> 1499, 1555, 1556, 1557, 1566, 1567, 1568, 1624, 1625, 1626, 1635, 1636, 1637	
<code>\dim_to_decimal_in_sp:n</code> 1288, 1293, 1298	
<code>\c_zero_dim</code> 1181, 1182, 1183, 1345, 1346, 1347	
<code>\directlua</code> 88, 1578, 1767	
E		
exp commands:		
<code>\exp_args:Ne</code> 1427, 1433, 1478, 1484, 1498, 1528, 1533, 1558, 1563, 1597, 1602, 1627, 1632	
<code>\exp_args:NNx</code> 874	
<code>\exp_args:Nnx</code> 482, 685, 709	
<code>\exp_args:Nxxx</code> 776, 791, 806, 821	
<code>\exp_args:Nx</code> 231, 342, 381, 681, 691, 705, 715, 770, 800, 1225	
<code>\exp_not:n</code> 608, 706, 801, 1410, 1431, 1482	
F		
fp commands:		
<code>\fp_eval:n</code> 1424, 1445, 1526, 1546, 1595, 1615	
G		
group commands:		
<code>\group_begin:</code> 1720	
<code>\group_end:</code> 1724	
H		
hbox commands:		
<code>\hbox:n</code> 1455, 1466	
<code>\hbox_gset:Nn</code> 1167	
<code>\hbox_set:Nn</code> 992, 1074, 1239, 1269, 1303, 1358	
hook commands:		
<code>\hook_gput_code:nnn</code>	.. 133, 470, 1348	
<code>\hook_gput_next_code:nn</code> 1184	
<code>\hook_gset_rule:nnnn</code> 464, 465	
I		
int commands:		
<code>\int_compare:nNnTF</code> 963, 1015, 1096, 1578, 1767	

<code>\int_compare_p:nNn</code>	<code>\l_pdf_current_structure_</code>
..... 1508, 1509, 1753, 1754	destination_tl
<code>\int_const:Nn</code> . 1048, 1128, 1163, 1265	1427, 1433, 1478, 1484, 1528, 1533,
<code>\int_gincr:N</code>	1558, 1563, 1597, 1602, 1627, 1632
204, 586, 605,	<code>\pdf_object_if_exist:nTF</code>
680, 704, 769, 773, 799, 803, 1162, 1264	.. 1427, 1478, 1528, 1558, 1597, 1627
<code>\int_if_exist:NTF</code>	<code>\pdf_object_new:n</code>
1382	419, 469
<code>\int_new:N</code>	<code>\pdf_object_ref:n</code> 868, 1029, 1034,
83, 84, 85	1039, 1044, 1109, 1114, 1119, 1124,
<code>\int_use:N</code>	1200, 1207, 1214, 1222, 1433, 1484
205, 208,	<code>\pdf_object_ref_last:</code> . 899, 906, 913
589, 597, 608, 616, 682, 687, 696,	<code>\pdf_object_unnamed_write:nn</code> ...
706, 711, 720, 771, 778, 782, 785,	... 630, 732, 834, 898, 905, 912, 1730
793, 801, 808, 812, 815, 823, 1056,	<code>\pdf_object_write</code>
1062, 1135, 1143, 1234, 1312, 1339,	485
1363, 1374, 1532, 1562, 1601, 1631	<code>\pdf_object_write:nnn</code>
	456, 473
	pdf internal commands:
K	<code>__pdf_backend:n</code>
kernel internal commands:	168,
<code>__kernel_backend_literal:n</code>	477, 486, 913, 978, 982, 1187, 1195,
..... 74, 587, 591, 606, 610, 624,	1196, 1203, 1210, 1217, 1226, 1241,
636, 657, 667, 1654, 1655, 1668, 1676	1408, 1429, 1457, 1458, 1468, 1480
<code>__kernel_backend_literal_page:n</code>	<code>__pdf_backend_bdc:nn</code>
..... 681, 705,	13,
728, 737, 759, 770, 800, 830, 839, 860	510, 522, 558, 639, 642, 643, 644,
<code>__kernel_backend_postscript:n</code> ..	740, 743, 744, 745, 842, 845, 846, 847
..... 1305, 1327, 1333, 1360	<code>__pdf_backend_bdc_contobj:nn</code> ...
<code>__kernel_backend_shipout_</code> 628, 642, 730, 743, 832, 845
<code>literal:n</code>	<code>__pdf_backend_bdc_contstream:nn</code>
27, 29, 531, 651 634, 643, 735, 744, 837, 846
<code>__kernel_backend_shipout_</code>	<code>__pdf_backend_bdc_shipout:nn</code> ...
<code>literal_page:n</code> ... 43, 43, 752, 853 529, 653, 754, 855
<code>__kernel_backend_shipout_</code>	<code>__pdf_backend_bdc_shipout_</code>
<code>literal_pdf:n</code>	contstream:nn
33, 33 649, 653, 750, 754, 851, 855
<code>__kernel_kern:n</code> 1454, 1462, 1465, 1494	<code>__pdf_backend_bdcobject:n</code>
<code>__kernel_pdf_name_from_unicode_</code> 13, 510,
<code>e:n</code>	540, 567, 603, 631, 702, 733, 797, 835
91, 97	<code>__pdf_backend_bdcobject:nn</code>
<code>__kernel_pdffdict_name:n</code> 13, 510, 536, 565, 584, 678, 767
..... 223, 224, 226,	<code>__pdf_backend_bmc:n</code>
454, 483, 665, 866, 877, 882, 995, 13, 510, 548, 571, 622, 726, 828
1016, 1027, 1032, 1037, 1042, 1077,	<code>__pdf_backend_catalog_gput:nn</code> .. 20
1097, 1107, 1112, 1117, 1122, 1272	<code>__pdf_backend_destination:nn</code> ...
<code>\g__kernel_pdfmanagement_end_</code> 1395, 1401
<code>run_code_tl</code>	<code>__pdf_backend_destination:nnnn</code> .
106, 113, 120 1396, 1402, 1550, 1619
<code>\g__kernel_pdfmanagement_</code>	<code>__pdf_backend_emc:</code>
<code>thispage_shipout_code_tl</code> 129, 135 13, 510, 544, 569, 655, 757, 858
	<code>__pdf_backend_link_begin:n</code> .. 1503
L	<code>__pdf_backend_link_begin:nnnw</code> ..
l ^a TeX commands: 1573, 1642
<code>\latelua:</code>	<code>__pdf_backend_link_begin_</code>
198, 277, 328, 367	goto:nnw
	1397, 1403
M	<code>__pdf_backend_link_begin_</code>
mode commands:	structure_goto:nnw
<code>\mode_leave_vertical:</code> ... 1186, 1350 1397, 1403, 1501, 1571, 1640
P	
pdf commands:	
<code>\pdf_activate_structure_destination:</code>	
..... 1392, 1393	

__pdf_backend_link_off:	__pdf_backend_ref_label:nn
. 946, 952, 965, 976 60, 70, 205, 782, 812
__pdf_backend_link_on:	__pdf_backend_ref_value:nn
. 947, 956, 969, 980 66, 71, 208, 785, 815
__pdf_backend_luastring:n	\g__pdf_backend_resourceid_int
. 154, 239, 248, 260, 261, 272, 287, 288 82, 204, 205, 208, 773, 778,
__pdf_backend_metadata_stream:n 782, 785, 793, 803, 808, 812, 815, 823
. 1711, 1718, 1728	__pdf_backend_set_regression_-
\g__pdf_backend_name_int	data: 1647
. 82, 586, 589, 597,	__pdf_backend_shipout_bdc:nn
605, 608, 616, 680, 682, 687, 696, 13, 510, 562
704, 706, 711, 720, 769, 771, 799, 801	__pdf_backend_structure_-
__pdf_backend_Names_gpush:nn	destination:nn
. 896, 903, 910, 919, 923 1395, 1401, 1405, 1406, 1511, 1580
__pdf_backend_NamesEmbeddedFiles_-	__pdf_backend_structure_-
add:nn 925, 926, 929, 941	destination:nnnn 1396, 1402, 1496
\g__pdf_backend_object_int	__pdf_backend_structure_-
. 1162, 1165, 1264, 1267, 1312	destination_aux:nnnn 1450, 1498
__pdf_backend_object_last:	__pdf_backend_ThisPage_gpush:n
. 542, 617, 712, 721, 809, 824 6, 175, 221, 297, 340, 379, 404
__pdf_backend_object_ref:n 426,	__pdf_backend_ThisPage_gput:nn
488, 538, 598, 670, 688, 697, 779, 794 6, 175, 201, 280, 331, 370, 401
__pdf_backend_object_write:nn	\g__pdf_backend_thispage_-
. 1714, 1723	shipout_tl 6
__pdf_backend_omit_charset:n	\l__pdf_backend_tmpa_box
. 1733, 1734, 1737, 1743 76, 992, 1001, 1004, 1007, 1047,
__pdf_backend_omit_info:n	1074, 1082, 1085, 1088, 1127, 1239,
. 1748, 1749, 1756, 1762, 1769, 1775	1246, 1247, 1248, 1249, 1269, 1278,
__pdf_backend_Page_gput:nn	1281, 1284, 1288, 1293, 1298, 1302,
. 6, 175, 185, 254, 315, 354, 390	1326, 1358, 1367, 1368, 1369, 1370
__pdf_backend_Page_gremove:n	\l__pdf_backend_tmpb_box
. 6, 175, 192, 268, 322, 361, 396 80, 1303, 1345, 1346, 1347, 1351
\g__pdf_backend_page_int 82	\l__pdf_backend_xform_bool
__pdf_backend_Page_primitive:n 513, 683,
. 6, 175, 178, 231,	707, 774, 804, 994, 1076, 1169, 1271
244, 308, 333, 342, 347, 372, 381, 387	__pdf_backend_xform_if_exist:n
__pdf_backend_PageResources:n 1380, 1386
. 475, 495, 503	__pdf_backend_xform_new:nnnn
\c__pdf_backend_PageResources_- 985, 986, 1068, 1156, 1254, 1262
clist 407, 417, 451, 467, 663, 888	__pdf_backend_xform_ref:n
__pdf_backend_PageResources_- 985, 1060,
gpush:n	1142, 1189, 1232, 1243, 1256, 1372
. 13, 510, 552, 573, 661, 762, 872	\l__pdf_backend_xform_tmpdp_tl
__pdf_backend_PageResources_- 1260, 1296, 1310, 1317
gpush_aux:n 863, 889	\l__pdf_backend_xform_tmplt_tl
__pdf_backend_PageResources_- 1261, 1291, 1315
gput:nnn 416, 432, 443, 479, 496, 504	\l__pdf_backend_xform_tmpwd_tl
__pdf_backend_PageResources_- 1259, 1286, 1316
obj_gpush: 416, 449, 492, 500, 508	__pdf_backend_xform_use:n
__pdf_backend_Pages_primitive:n 985, 1053, 1133, 1237, 1255, 1356
. 140, 141, 150, 160, 166, 172	\g__pdf_tmpa_prop 76, 223, 228, 233
__pdf_backend_pdfmark:n	\l__pdf_tmpa_tl
. 524, 538, 542, 546, 550, 931 76, 206, 210, 212, 215, 783,
	787, 789, 792, 813, 817, 819, 822, 825

pdfdict commands:	
\pdfdict_gput:nnn	187, 215, 317, 356, 392, 434, 445, 498, 506, 685, 709, 776, 791, 806, 821
\pdfdict_gremove:nn	194, 324, 363, 398
\pdfdict_if_exist:nTF	210, 787, 817
\pdfdict_item:mn	233, 868, 883
\pdfdict_new:n	212, 789, 819
\pdfdict_show:n	825
\pdfdict_use:n	343, 382, 458, 1022, 1103
\pdfextension	967, 971
\pdfliteral	1
pdfmanagement commands:	
\pdfmanagement_add:nnn	1650, 1653, 1658, 1659, 1662, 1663, 1667, 1675, 1683, 1688, 1699, 1700
pdfmanagement internal commands:	
\g_pdfmanagement_active_bool	641, 742, 844
\l_pdfmanagement_delayed_	
shipout_bool	27, 527, 560, 647, 748, 849
\pdfnames	20
\pdfomitinfodict	1758
\pdfpageref	3
\pdfrunninglinkoff	950, 954
\pdfrunninglinkon	958
\pdftrailerid	1685
pdfxform commands:	
\pdfxform_dp:n	1192, 1248, 1369
\pdfxform_ht:n	1191, 1247, 1368
\pdfxform_if_exist:n	1386
\pdfxform_wd:n	1190, 1246, 1367
prg commands:	
\prg_new_conditional:Npnn	1380
\prg_new_eq_conditional:NNn	1386
\prg_return_false:	1384
\prg_return_true:	1383
prop commands:	
\prop_count:N	1016, 1097
\prop_gclear:N	995, 1077, 1272
\prop_gput:Nnn	228, 483
\prop_gset_eq:NN	223
\prop_if_empty:NTF	453, 665, 865, 1026, 1031, 1036, 1041, 1106, 1111, 1116, 1121
\prop_if_exist:NTF	224, 876
\prop_map_function:NN	233, 881
\prop_map_inline:Nn	226
\prop_new:N	77
\ProvidesExplFile	1
	R
ref commands:	
\ref_label:nn	58, 63
\ref_value:nn	59, 68
\relax	126, 1689
\RequirePackage	57
	S
scan commands:	
\scan_stop:	1057, 1139, 1527, 1547, 1557, 1568, 1596, 1616, 1626, 1637, 1684, 1721, 1739, 1745, 1758, 1771
\special	1
str commands:	
\str_case:nnTF	1413, 1434, 1515, 1535, 1584, 1604
\str_convert_pdfname:n	93, 484
\str_if_eq:nnTF	1320, 1331
\str_if_exist:NTF	1656, 1697
sys commands:	
\c_sys_engine_exec_str	1707
\sys_gset_rand_seed:n	1649
\sys_if_engine luatex:TF	148
\c_sys_timestamp_str	1656, 1658, 1659, 1697
	T
T _E X and L ^A T _E X 2 _ε commands:	
\@bsphack	62
\@esphack	64
\@kernel@after@enddocument@afterlastpage	103, 104
\@kernel@after@shipout@background	124, 127
\@kernel@after@shipout@lastpage	110, 111, 117, 118
\@kernel@before@shipout@background	126
\g@addto@macro	126, 127
\special	2
tex commands:	
\tex_directlua:D	152, 256, 270, 420, 422
\tex_global:D	143, 180, 874
\tex_immediate:D	1009, 1090, 1713, 1722
\tex_latelua:D	246, 282, 299, 435, 436, 691, 715
\tex_luaescapestring:D	241
\tex luatexversion:D	963
\tex_pdfcompresslevel:D	1721
\tex_pdfdest:D	1513, 1530, 1552, 1560
\tex_pdfextension:D	36, 46, 906, 1582, 1599, 1621, 1629, 1713
\tex_pdflastxform:D	1050, 1130
\tex_pdfliteral:D	39, 49

<code>\tex_pdfnames:D</code>	899	tl commands:	
<code>\tex_pdfobj:D</code>	1722	<code>\c_space_tl</code>	589, 597, 608, 616, 682, 706, 771, 801, 1190, 1191, 1192, 1312
<code>\tex_pdfomitchar:D</code>	1739	<code>\tl_const:Nn</code>	999, 1002, 1005, 1080, 1083, 1086, 1172, 1175, 1178, 1276, 1279, 1282
<code>\tex_pdfpageattr:D</code>	180	<code>\tl_gput_right:Nn</code>	104, 111, 118
<code>\tex_pdfpageresources:D</code>	874	<code>\tl_if_exist:NTF</code>	124
<code>\tex_pdfpagesattr:D</code>	143	<code>\tl_new:N</code>	78, 1259, 1260, 1261, 1390
<code>\tex_pdfrefxform:D</code>	1055, 1135	<code>\tl_set:Nn</code>	206, 783, 813, 1286, 1291, 1296
<code>\tex_pdfsuppressptexinfo:D</code>	1684	<code>\tl_to_str:n</code>	1000, 1003, 1006, 1049, 1056, 1062, 1081, 1084, 1087, 1129, 1137, 1143, 1164, 1173, 1176, 1179, 1234, 1266, 1277, 1280, 1283, 1339, 1363, 1374, 1382, 1533, 1563, 1602, 1632
<code>\tex_pdftexrevision:D</code>	1509, 1754	<code>\tl_use:N</code>	1310, 1315, 1316, 1317
<code>\tex_pdftexversion:D</code>	1508, 1753		
<code>\tex_pdfvariable:D</code>	1689, 1690, 1745, 1771		
<code>\tex_pdfxform:D</code>	1009, 1090		
<code>\tex_special:D</code>	30, 162, 310, 349		
<code>\tex_the:D</code>	1001, 1004, 1007, 1082, 1085, 1088, 1174, 1177, 1180, 1278, 1281, 1284		
<code>\tex_unexpanded:D</code>	241		
<code>\tex_vss:D</code>	1460, 1492		
text commands:			
<code>\text_expand:n</code>	93, 99	vbox commands:	
		<code>\vbox_to_zero:n</code>	1452, 1463

V