

The l3backend-testphase package

Additional backend PDF features

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.96b, released 2023-11-05

1 l3backend-testphase Implementation

```
1 <drivers>\ProvidesExplFile
2 <*dvipdfmx>
3   {l3backend-testphase-dvipdfmx.def}{2023-11-05}{ }
4   {LaTeX~PDF~management~testphase~bundle~backend~support: dvipdfmx}
5 </dvipdfmx>
6 <*dvips>
7   {l3backend-testphase-dvips.def}{2023-11-05}{ }
8   {LaTeX~PDF~management~testphase~bundle~backend~support: dvips}
9 </dvips>
10 <*dvisvgm>
11   {l3backend-testphase-dvisvgm.def}{2023-11-05}{ }
12   {LaTeX~PDF~management~testphase~bundle~backend~support: dvisvgm}
13 </dvisvgm>
14 <*luatex>
15   {l3backend-testphase-luatex.def}{2023-11-05}{ }
16   {LaTeX~PDF~management~testphase~bundle~backend~support: PDF output (LuaTeX)}
17 </luatex>
18 <*pdftex>
19   {l3backend-testphase-pdftex.def}{2023-11-05}{ }
20   {LaTeX~PDF~management~testphase~bundle~backend~support: PDF output (pdfTeX)}
21 </pdftex>
22 <*xdvipdfmx>
23   {l3backend-testphase-xetex.def}{2023-11-05}{ }
24   {LaTeX~PDF~management~testphase~bundle~backend~support: XeTeX}
25 </xdvipdfmx>
```

1.1 Variants

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

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

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

```

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

```

1.2 Support for delayed literal and special

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

```

38 <*drivers>

```

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

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

```

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

```

(End of definition for __kernel_backend_shipout_literal:e.)

```

44 <*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).

```

45 \cs_new_protected:Npn \__kernel_backend_shipout_literal_pdf:e #1
46 {
47 <*luatex>
48   \tex_pdfextension:D ~ literal ~ shipout ~
49 </luatex>
50 <*pdftex>
51   \tex_pdfliteral:D ~ shipout ~
52 </pdftex>
53   { #1 }
54 }

```

(End of definition for __kernel_backend_shipout_literal_pdf:e.)

`__kernel_backend_shipout_literal_page:e` Page literals are pretty simple.

```

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

```

```

60 <*pdftex>
61     \tex_pdfliteral:D ~ shipout ~
62 </pdftex>
63     page { #1 }
64 }
65 </luatex | pdftex>
66 <drivers> }

```

(End of definition for __kernel_backend_shipout_literal_page:e.)

1.3 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/ltshipout package.

```

67 <*drivers>

```

l3ref-tmp is loaded until all files in latex-lab and tagpdf has been cleared:

```

68 \RequirePackage{l3ref-tmp}
69 \cs_if_exist:NTF \property_new:nnnn
70 {
71     \cs_new_protected:Npn \__pdf_backend_record_abspage:n #1
72     {
73         \@bsphack
74         \property_record:nn{#1}{abspage}
75         \@esphack
76     }
77     \cs_new:Npn \__pdf_backend_ref_abspage:n #1
78     {
79         \property_ref:nn{#1}{abspage}
80     }
81 }
82 {
83     {
84         \cs_new_protected:Npn \__pdf_backend_record_abspage:n #1
85         %% \__pdf_backend_ref_label:nn #1 #2
86         {
87             \@bsphack
88             \ref_label:nn{#1}{abspage}
89             \@esphack
90         }
91         \cs_new:Npn \__pdf_backend_ref_abspage:n #1
92         %% \__pdf_backend_ref_value:nn #1 #2
93         {
94             \ref_value:nn{#1}{abspage}
95         }
96     }
97     \cs_generate_variant:Nn \__pdf_backend_record_abspage:n {e}
98     \cs_generate_variant:Nn \__pdf_backend_ref_abspage:n {e}
99 </drivers>

```

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

```

100 <*dvipdfmx | xdvipdfmx>

```

```

101     \__kernel_backend_literal:n { dvipdfmx:config~C~ 0x0010 }
102 </dvipdfmx | xdvipdfmx>

```

```

\g__pdf_tmpa_prop
\l__pdf_tmpa_tl
\l__pdf_backend_tmpa_box

```

Some scratch variables

```

103 <*drivers>
104 \prop_new:N \g__pdf_tmpa_prop
105 \tl_new:N \l__pdf_tmpa_tl
106 \box_new:N \l__pdf_backend_tmpa_box
107 \box_new:N \l__pdf_backend_tmpb_box
108 </drivers>

```

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

```

\g__pdf_backend_resourceid_int
\g__pdf_backend_name_int
\g__pdf_backend_page_int

```

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

```

109 <*drivers>
110 \int_new:N \g__pdf_backend_resourceid_int
111 \int_new:N \g__pdf_backend_name_int
112 \int_new:N \g__pdf_backend_page_int
113 </drivers>

```

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

1.4 luacode

Load the lua code.

```

114 <*luatex>
115     \directlua { require("l3backend-testphase.lua") }
116 </luatex>

```

1.5 Converting unicode strings to a pdfname

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

```

117 <*pdfTeX | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
118 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
119 {
120     / \str_convert_pdfname:e { \text_expand:n { #1 } }
121 }
122 </pdfTeX | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
123 <*dvips>
124 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
125 {
126     ~ ( \text_expand:n { #1 } ) ~ cvn
127 }
128 </dvips>

```

1.6 Hooks

1.6.1 Add the “end run” hooks

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

```
129 <*pdftex | luatex>
130 % put in \@kernel@after@enddocument@afterlastpage
131 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
132 {
133   \g__kernel_pdfmanagement_end_run_code_tl
134 }
135 </pdftex | luatex>
136 <*dvipdfmx | xdvipdfmx>
137 % put in \@kernel@after@shipout@lastpage
138 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
139 {
140   \g__kernel_pdfmanagement_end_run_code_tl
141 }
142 </dvipdfmx | xdvipdfmx>
143 <*dvips>
144 % put in \@kernel@after@shipout@lastpage
145 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
146 {
147   \g__kernel_pdfmanagement_end_run_code_tl
148 }
149 </dvips>
```

1.6.2 Add the “shipout” hooks

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

```
150 <*drivers>
151 \tl_if_exist:NTF \@kernel@after@shipout@background
152 {
153   \g@addto@macro \@kernel@before@shipout@background{\relax}
154   \g@addto@macro \@kernel@after@shipout@background
155   {
156     \g__kernel_pdfmanagement_thispage_shipout_code_tl
157   }
158 }
159 {
160   \hook_gput_code:nnn{shipout/background}{pdf}
161   {
162     \g__kernel_pdfmanagement_thispage_shipout_code_tl
163   }
164 }
165
166 </drivers>
```

1.7 The /Pages dictionary (pdfpagesattr)

_pdf_backend_Pages_primitive:n

This is the primitive command to add something to the /Pages dictionary. It works differently for the backends: pdftex and luatex overwrite existing content, dvips and

dvipdfmx are additive. luatex sets it in lua. The higher level code has to take this into account.

```

167 < *pdftex >
168 \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
169 {
170   \tex_global:D \tex_pdfpagesattr:D { #1 }
171 }
172 < /pdftex >
173 < *luatex >
174 %luatex: does it in lua
175 \sys_if_engine luatex:T
176 {
177   \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
178   {
179     \tex_directlua:D
180     {
181       pdf.setpagesattributes( \__pdf_backend_luastring:n { #1 } )
182     }
183   }
184 }
185 < /luatex >
186 < *dvips >
187 \cs_new_protected:Npx \__pdf_backend_Pages_primitive:n #1
188 {
189   \tex_special:D{ps:~[#1~/PAGES~pdfmark} %]
190 }
191 < /dvips >
192 < *dvipdfmx | xdvipdfmx >
193 \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
194 {
195   \__pdf_backend:n{put~@pages~<<#1>>}
196 }
197 < /dvipdfmx | xdvipdfmx >
198 < *dvisvgm >
199 \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
200 {}
201 < /dvisvgm >

```

(End of definition for __pdf_backend_Pages_primitive:n.)

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

```

\__pdf_backend_Page_primitive:n
\__pdf_backend_Page_gput:nn
\__pdf_backend_Page_gremove:n
\__pdf_backend_ThisPage_gput:nn
\__pdf_backend_ThisPage_gpush:n

```

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

```

202 % backend commands
203 < *pdftex >
204 %the primitive

```

```

205 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
206 {
207   \tex_global:D \tex_pdfpageattr:D { #1 }
208 }
209 % the command to store default values.
210 % Uses a prop with pdflatex + dvi,
211 % sets a lua table with luatex
212 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2 %key,value
213 {
214   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
215 }
216 % the command to remove a default value.
217 % Uses a prop with pdflatex + dvi,
218 % changes a lua table with luatex
219 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
220 {
221   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
222 }
223 % the command used in the document.
224 % direct call of the primitive special with dvips/dvipdfmx
225 % \latelua: fill a page related table with luatex, merge it with the page
226 % table and push it directly
227 % write to aux and store in prop with pdflatex
228 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
229 {
230   %we need to know the page the resource should be added too.
231   \int_gincr:N\g__pdf_backend_resourceid_int
232   \__pdf_backend_record_abspage:e { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
233   \tl_set:Ne \l__pdf_tmpa_tl
234   {
235     \__pdf_backend_ref_abspage:e {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
236   }
237   \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
238   {
239     \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
240   }
241   %backend_Page has no handler.
242   \pdfdict_gput:nnn {g__pdf_Core/backend_Page\l__pdf_tmpa_tl}{ #1 }{ #2 }
243 }
244 %the code to push the values, used in shipout
245 %merges the two props and then fills the register in pdflatex
246 %merges the two tables and then fills (in lua) in luatex
247 %issues the values stored in the global prop with dvi
248 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
249 {
250   \prop_gset_eq:Nc \g__pdf_tmpa_prop { \__kernel_pdfdict_name:n { g__pdf_Core/Page } }
251   \prop_if_exist:cT { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
252   {
253     \prop_map_inline:cn { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
254     {
255       \prop_gput:Nnn \g__pdf_tmpa_prop { ##1 }{ ##2 }
256     }
257   }
258   \__pdf_backend_Page_primitive:e

```

```

259     {
260         \prop_map_function:NN \g__pdf_tmpa_prop \pdfdict_item:ne
261     }
262 }
263 </pdfTeX>
264 <*luatex>
265 % do we need to use some escaping for the values????
266 \cs_new:Npn \__pdf_backend_luastring:n #1
267 {
268     "\tex_luaescapestring:D { \tex_unexpanded:D { #1 } }"
269 }
270 %not used, only there for consistency
271 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
272 {
273     \tex_latelua:D
274     {
275         pdf.setpageattributes(\__pdf_backend_luastring:n { #1 })
276     }
277 }
278 % the command to store default values.
279 % Uses a prop with pdfLaTeX + dvi,
280 % sets a lua table with LuaLaTeX
281 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
282 {
283     \tex_directlua:D
284     {
285         ltx.__pdf.backend_Page_gput
286         (
287             \__pdf_backend_luastring:n { #1 },
288             \__pdf_backend_luastring:n { #2 }
289         )
290     }
291 }
292 % the command to remove a default value.
293 % Uses a prop with pdfLaTeX + dvi,
294 % changes a lua table with LuaLaTeX
295 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
296 {
297     \tex_directlua:D
298     {
299         ltx.__pdf.backend_Page_gremove (\__pdf_backend_luastring:n { #1 })
300     }
301 }
302 % the command used in the document.
303 % direct call of the primitive special with dvips/dvipdfmx
304 % \latelua: fill a page related table with LuaLaTeX, merge it with the page
305 % table and push it directly
306 % write to aux and store in prop with pdfLaTeX
307 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
308 {
309     \tex_latelua:D
310     {
311         ltx.__pdf.backend_ThisPage_gput
312         (

```

```

313         tex.count["g_shipout_readonly_int"],
314         \__pdf_backend_luastring:n { #1 },
315         \__pdf_backend_luastring:n { #2 }
316     )
317     ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
318 }
319 }
320 %the code to push the values, used in shipout
321 %merges the two props and then fills the register in pdflatex
322 %merges the two tables (the one is probably still empty) and then fills (in lua) in luatex
323 %issues the values stored in the global prop with dvi
324 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
325 {
326     \tex_latelua:D
327     {
328         ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
329     }
330 }
331
332 </luatex>
333 <*dvipdfmx|xdvipdfmx>
334 %the primitive
335 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
336 {
337     \tex_special:D{pdf:-put~@thispage-<<#1>>}
338 }
339 % the command to store default values.
340 % Uses a prop with pdflatex + dvi,
341 % sets a lua table with luatex
342 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
343 {
344     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
345 }
346 % the command to remove a default value.
347 % Uses a prop with pdflatex + dvi,
348 % changes a lua table with luatex
349 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
350 {
351     \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
352 }
353 % the command used in the document.
354 % direct call of the primitive special with dvips/dvipdfmx
355 % \latelua: fill a page related table with luatex, merge it with the page
356 % table and push it directly
357 % write to aux and store in prop with pdflatex
358 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
359 {
360     \__pdf_backend_Page_primitive:n { /#1~#2 }
361 }
362 %the code to push the values, used in shipout
363 %merges the two props and then fills the register in pdflatex
364 %merges the two tables (the one is probably still empty)
365 % and then fills (in lua) in luatex
366 %issues the values stored in the global prop with dvi

```

```

367 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
368 {
369   \__pdf_backend_Page_primitive:e
370   { \pdfdict_use:n { g__pdf_Core/Page} }
371 }
372 </dvipdfmx | xdvipdfmx>
373 <*dvips>
374 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
375 {
376   \tex_special:D{ps:~[{\ThisPage}<<#1>>~/PUT~pdfmark} %]
377 }
378 % the command to store default values.
379 % Uses a prop with pdflatex + dvi,
380 % sets a lua table with luatex
381 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
382 {
383   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
384 }
385 % the command to remove a default value.
386 % Uses a prop with pdflatex + dvi,
387 % changes a lua table with luatex
388 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
389 {
390   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
391 }
392 % the command used in the document.
393 % direct call of the primitive special with dvips/dvipdfmx
394 % \latelua: fill a page related table with luatex, merge it with the page
395 % table and push it directly
396 % write to aux and store in prop with pdflatex
397 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
398 {
399   \__pdf_backend_Page_primitive:n { /#1~#2 }
400 }
401 %the code to push the values, used in shipout
402 %merges the two props and then fills the register in pdflatex
403 %merges the two tables (the one is probably still empty)
404 %and then fills (in lua) in luatex
405 %issues the values stored in the global prop with dvi
406 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
407 {
408   \__pdf_backend_Page_primitive:e
409   { \pdfdict_use:n { g__pdf_Core/Page} }
410 }
411 </dvips>
412 <*dvisvgm>
413 % mostly only dummies ...
414 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
415 {}
416 % Uses a prop with pdflatex + dvi,
417 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
418 {
419   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
420 }

```

```

421 % the command to remove a default value.
422 % Uses a prop with pdflatex + dvi,
423 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
424 {
425   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
426 }
427 % the command used in the document.
428 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
429 {}
430 %the code to push the values, used in shipout
431 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
432 {}
433 </divisvgm>
434 <*drivers>
435 \cs_generate_variant:Nn \__pdf_backend_Page_primitive:n { e }
436 </drivers>

```

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

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

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

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

```

437 <*drivers>
438 \clist_const:Nn \c__pdf_backend_PageResources_clist
439 {
440   ExtGState,
441   ColorSpace,
442   Pattern,
443   Shading,
444 }
445 </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

__pdf_backend_PageResources_obj_gpush: 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:

```

446 <*pdfTeX | luatex>
447 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
448 {
449   \pdf_object_new:n {__pdf/Page/Resources/#1}
450   \cs_if_exist:NT \tex_directlua:D
451   {
452     \tex_directlua:D

```

```

453         {
454             ltx.__pdf.object["__pdf/Page/Resources/#1"]
455             =
456             "\\__pdf_backend_object_ref:n{__pdf/Page/Resources/#1}"
457         }
458     }
459 }
460 </pdftex | luatex>

```

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

```

461 <*luatex>
462 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
463 {
464     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
465     \tex_latelua:D{ltx.__pdf.Page.Resources.#1=true}
466     \tex_latelua:D
467     {
468         ltx.pdf.Page.Resources_gpush(tex.count["g_shipout_readonly_int"])
469     }
470 }
471 </luatex>
472 <*pdftex>
473 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
474 {
475     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
476 }
477 </pdftex>

```

code for end of document code

```

478 <*pdftex | luatex>
479 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush:
480 {
481     \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
482     {
483         \prop_if_empty:cF
484         { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1 } }
485         {
486             \pdf_object_write:nne
487             { __pdf/Page/Resources/##1 } { dict }
488             { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1 } }
489         }
490     }
491 }
492 </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.

```

493 <*dvipdfmx | xdvipdfmx>
494 <xdvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-xetex}{after}{pdf}
495 <dvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-dvipdfmx}{after}{pdf}
496 %

```

```

497 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
498 {
499   \pdf_object_new:n { __pdf/Page/Resources/#1 }
500   \hook_gput_code:nnn
501     {shipout/firstpage}
502     {pdf}
503     {\pdf_object_write:nnn { __pdf/Page/Resources/#1 } { dict } {}}
504 }
505 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1
506 {
507   \__pdf_backend:n {put~@resources~<<#1>>}
508 }
509 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
510 {
511   % this is not used for output, but there is a test if the resource is empty
512   \prop_gput:cne { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1 } }
513     { \str_convert_pdfname:n {#2} }{ #3 }
514   %objects are not filled with \pdf_object_write as this is not additive!
515   \__pdf_backend:e
516     {
517       put~\__pdf_backend_object_ref:n {__pdf/Page/Resources/#1}<</#2~#3>>
518     }
519 }
520
521 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
522 </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.

```

523 <*dvips>
524 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
525 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
526 { %only for the show command TEST!!
527   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
528 }
529 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
530 </dvips>

```

dvipsvgm unneeded, or no-op

```

531 <*dvisvgm>
532 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
533 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
534 { %only for the show command TEST!!
535   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
536 }
537 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
538 </dvisvgm>

```

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

1.9.1 Page resources /Properties + BDC operators

```

\__pdf_backend_bdc:nn \__pdf_backend_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
\__pdf_backend_bdcobject:n
\__pdf_backend_bmc:n
\__pdf_backend_emc:
\__pdf_backend_PageResources_gpush:n

```

the backend command that create the bdc/emc marker and store the properties. `_pdf_backend_PageResources_gpush:n` outputs the `/Properties` and/or the other resources for the current page.

```

539 % pdftex and luatex (and perhaps dvips ...) need to know if there are in a
540 % xform stream ...
541 <*drivers>
542 \bool_new:N \l__pdf_backend_xform_bool
543 </drivers>
544 <*dvips>
545 % dvips is easy: create an object, and reference it in the bdc
546 % ghostscript will then automatically replace it by a name
547 % and add the name to the /Properties dict
548 % special variant von accsupp
549 % https://chat.stackexchange.com/transcript/message/50831812#50831812
550 %
551 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
552 {
553   \__pdf_backend_pdfmark:n{/#1~<<#2>>~/BDC}
554 }
555
556 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
557 {
558   \cs_new_protected:Npn \__pdf_backend_bdc_shipout:ee #1 #2 % #1 eg. Span, #2: dict_content
559   {
560     \__kernel_backend_shipout_literal:e
561     {ps: SDict ~ begin ~ mark /#1~<<#2>>~/BDC ~ pdfmark ~ end }
562   }
563 }
564
565 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
566 {
567   \__pdf_backend_pdfmark:e{/#1~\__pdf_backend_object_ref:n{#2}~/BDC}
568 }
569 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
570 {
571   \__pdf_backend_pdfmark:e{/#1~\__pdf_backend_object_last:~/BDC}
572 }
573 \cs_set_protected:Npn \__pdf_backend_emc:
574 {
575   \__pdf_backend_pdfmark:n{EMC} %
576 }
577 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
578 {
579   \__pdf_backend_pdfmark:n{/#1~/BMC} %
580 }
581 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
582
583 </dvips>
584 <*dvisvgm>
585 % dvisvgm should do nothing
586 %
587 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
588 {}
589 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool

```

```

590 {
591   \cs_set_protected:Npn \__pdf_backend_shipout_bdc:ee #1 #2 % #1 eg. Span, #2: dict_content
592   {}
593 }
594 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
595 {}
596 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
597 {}
598 \cs_set_protected:Npn \__pdf_backend_emc:
599 {}
600 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
601 {}
602 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
603
604 </divisvgm>
605
606 % xetex has to create the entries in the /Properties manually
607 % (like the other backends)
608 % use pdfbase special
609 % https://chat.stackexchange.com/transcript/message/50832016#50832016
610 % the property is added to xform resources automatically,
611 % no need to worry about it.
612 <*dvipdfmx | xdvipdfmx>
613 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
614 {
615   \int_gincr:N \g__pdf_backend_name_int
616   \__kernel_backend_literal:e
617   {
618     pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
619   }
620   \__kernel_backend_literal:e
621   {
622     pdf:put~@resources~
623     <<
624       /Properties~
625       <<
626         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
627         \__pdf_backend_object_ref:n { #2 }
628       >>
629     >>
630   }
631 }
632 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span
633 {
634   \int_gincr:N \g__pdf_backend_name_int
635   \__kernel_backend_literal:e
636   {
637     pdf:code~/\exp_not:n{#1}/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
638   }
639   \__kernel_backend_literal:e
640   {
641     pdf:put~@resources~
642     <<
643     /Properties~

```

```

644         <<
645         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
646         \__pdf_backend_object_last:
647         >>
648     >>
649 }
650 }
651 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
652 {
653     \__kernel_backend_literal:n {pdf:code~/#1~BMC} %pdfbase
654 }
655
656 %this require management
657 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
658 {
659     \pdf_object_unnamed_write:nn { dict }{ #2 }
660     \__pdf_backend_bdcobject:n { #1 }
661 }
662
663 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
664 {
665     \__kernel_backend_literal:n {pdf:code~ /#1~<<#2>>~BDC }
666 }
667
668 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
669 {
670     \bool_if:NTF \g__pdfmanagement_active_bool
671     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
672     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
673     \__pdf_backend_bdc:nn {#1}{#2}
674 }
675
676 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
677 {
678     \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
679     {
680         \__kernel_backend_shipout_literal:e {pdf:code~ /#1~<<#2>>~BDC }
681     }
682     \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
683 }
684 \cs_set_protected:Npn \__pdf_backend_emc:
685 {
686     \__kernel_backend_literal:n {pdf:code~EMC} %pdfbase
687 }
688 % properties are handled automatically, but the other resources should be added
689 % at shipout
690 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
691 {
692     \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
693     {
694         \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
695         {
696             \__kernel_backend_literal:e
697             {

```

```

698         pdf:put~@resources~
699         <</##1~\__pdf_backend_object_ref:n {__pdf/Page/Resources/##1}>>
700     }
701 }
702 }
703 }
704 </dviptdmx | xdvipdmx>
705 % luatex + pdftex
706 <*luatex>
707 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
708 {
709     \int_gincr:N \g__pdf_backend_name_int
710     \__kernel_backend_literal_page:e
711     { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
712     \bool_if:NTF \l__pdf_backend_xform_bool
713     {
714         \pdfdict_gput:nee
715         { g__pdf_Core/Xform/Resources/Properties }
716         { l3pdf\int_use:N\g__pdf_backend_name_int }
717         { \__pdf_backend_object_ref:n { #2 } }
718     }
719     {
720         \exp_args:Ne \tex_latelua:D
721         {
722             ltx.pdf.Page_Resources_Properties_gput
723             (
724                 tex.count["g_shipout_readonly_int"],
725                 "l3pdf\int_use:N\g__pdf_backend_name_int",
726                 "\__pdf_backend_object_ref:n { #2 }"
727             )
728         }
729     }
730 }
731 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
732 {
733     \int_gincr:N \g__pdf_backend_name_int
734     \__kernel_backend_literal_page:e
735     { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
736     \bool_if:NTF \l__pdf_backend_xform_bool
737     {
738         \pdfdict_gput:nee %no handler needed
739         { g__pdf_Core/Xform/Resources/Properties }
740         { l3pdf\int_use:N\g__pdf_backend_name_int }
741         { \__pdf_backend_object_last: }
742     }
743     {
744         \exp_args:Ne \tex_latelua:D
745         {
746             ltx.pdf.Page_Resources_Properties_gput
747             (
748                 tex.count["g_shipout_readonly_int"],
749                 "l3pdf\int_use:N\g__pdf_backend_name_int",
750                 "\__pdf_backend_object_last:"
751             )

```

```

752     }
753   }
754 }
755 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
756 {
757   \__kernel_backend_literal_page:n { /#1~BMC }
758 }
759 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
760 {
761   \pdf_object_unnamed_write:nn { dict } { #2 }
762   \__pdf_backend_bdcobject:n { #1 }
763 }
764 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
765 {
766   \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
767 }
768
769 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
770 {
771   \bool_if:NTF \g__pdfmanagement_active_bool
772     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
773     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
774     \__pdf_backend_bdc:nn {#1}{#2}
775 }
776
777 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
778 {
779   \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
780   {
781     \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
782   }
783   \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
784 }
785
786 \cs_set_protected:Npn \__pdf_backend_emc:
787 {
788   \__kernel_backend_literal_page:n { EMC }
789 }
790
791 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
792 </luatex>
793 <*pdftex>
794 % pdfflatex is the most complicated as it has to go through the aux ...
795 % the push command is extended to take other resources too
796 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
797 {
798   \int_gincr:N \g__pdf_backend_name_int
799   \__kernel_backend_literal_page:e
800   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
801   % code to set the property ...
802   \int_gincr:N \g__pdf_backend_resourceid_int
803   \bool_if:NTF \l__pdf_backend_xform_bool
804   {
805     \pdfdict_gput:nee %no handler needed

```

```

806         { g__pdf_Core/Xform/Resources/Properties }
807         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
808         { \__pdf_backend_object_ref:n { #2 } }
809     }
810     {
811         \__pdf_backend_record_abspage:e {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
812         \tl_set:N\l__pdf_tmpa_tl
813         {
814             \__pdf_backend_ref_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
815         }
816         \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
817         {
818             \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
819         }
820         \pdfdict_gput:nee
821         { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
822         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
823         { \__pdf_backend_object_ref:n{#2} }
824     }
825 }
826 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
827 {
828     \int_gincr:N \g__pdf_backend_name_int
829     \__kernel_backend_literal_page:e
830     { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
831     % code to set the property ....
832     \int_gincr:N\g__pdf_backend_resourceid_int
833     \bool_if:NTF \l__pdf_backend_xform_bool
834     {
835         \pdfdict_gput:nee
836         { g__pdf_Core/Xform/Resources/Properties }
837         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
838         { \__pdf_backend_object_last: }
839     }
840     {
841         \__pdf_backend_record_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
842         \tl_set:N\l__pdf_tmpa_tl
843         {
844             \__pdf_backend_ref_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
845         }
846         \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
847         {
848             \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
849         }
850         \pdfdict_gput:nee
851         { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
852         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
853         { \__pdf_backend_object_last: }
854         %\pdfdict_show:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
855     }
856 }
857 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
858 {
859     \__kernel_backend_literal_page:n { /#1-BMC }

```

```

860 }
861 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
862 {
863   \pdf_object_unnamed_write:nn { dict } { #2 }
864   \__pdf_backend_bdcobject:n { #1 }
865 }
866 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
867 {
868   \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
869 }
870
871 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
872 {
873   \bool_if:NTF \g_pdfmanagement_active_bool
874     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
875     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
876     \__pdf_backend_bdc:nn {#1}{#2}
877 }
878 \bool_if:NT\l_pdfmanagement_delayed_shipout_bool
879 {
880   \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
881   {
882     \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
883   }
884   \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
885 }
886
887 \cs_set_protected:Npn \__pdf_backend_emc:
888 {
889   \__kernel_backend_literal_page:n { EMC }
890 }
891
892 \cs_new:Npn \__pdf_backend_PageResources_gpush_aux:n #1 %#1 ExtGState etc
893 {
894   \prop_if_empty:cF
895     { \__kernel_pdffdict_name:n { g_pdf_Core/Page/Resources/#1 } }
896     {
897       \pdffdict_item:ne { #1 }{ \pdf_object_ref:n {__pdf/Page/Resources/#1}}
898     }
899 }
900
901 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
902 {
903   \exp_args:NNe \tex_global:D \tex_pdfpageresources:D
904   {
905     \prop_if_exist:cT
906       { \__kernel_pdffdict_name:n { g_pdf_Core/backend_Page#1/Resources/Properties } }
907       {
908         /Properties~
909         <<
910         \prop_map_function:cN
911           { \__kernel_pdffdict_name:n { g_pdf_Core/backend_Page#1/Resources/Property
912           \pdffdict_item:ne
913           >>

```

```

914     }
915     %% add ExtGState etc
916     \clist_map_function:NN
917       \c__pdf_backend_PageResources_clist
918       \__pdf_backend_PageResources_gpush_aux:n
919   }
920 }
921
922 </pdftex>

```

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

1.10 “Catalog” & subdirectories (pdfcatalog)

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

1.10.1 Special case: the /Names/EmbeddedFiles dictionary

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

```

923 % pdflatex
924 <*pdftex>
925 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 % #1 name of name tree, #2 array co
926 {
927   \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
928   \tex_pdfnames:D {/#1~\pdf_object_ref_last:}
929 }
930 </pdftex>
931 <*luatex>
932 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 % #1 name of name tree, #2 array co
933 {
934   \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
935   \tex_pdfextension:D~names~ {/#1~\pdf_object_ref_last:}
936 }
937 </luatex>
938 <*dviptdpmx | xdvipdpmx>
939 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 % #1 name of name tree, #2 array co
940 {
941   \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
942   \__pdf_backend:e {put~@names~<</#1~\pdf_object_ref_last: >>}
943 }
944 </dviptdpmx | xdvipdpmx>
945
946 %dvips: noop
947 <*dvips>
948 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
949 </dvips>
950 %dvisvgm: noop
951 <*dvisvgm>
952 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
953 </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.

```

954 <*pdftex | luatex | dvipdfmx | xdvipdfmx>
955 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2 {}
956 </pdftex | luatex | dvipdfmx | xdvipdfmx>
957 <*dvips>
958 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
959 {
960     \__pdf_backend_pdfmark:e
961     {
962         /Name~#1~
963         /FS~#2~
964         /EMBED
965     }
966 }
967 </dvips>
968 <*dvisvgm>
969 %no op. Or is there any sensible use for it?
970 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
971 {}
972
973 </dvisvgm>

```

(End of definition for __pdf_backend_NamesEmbeddedFiles_add:nn.)

1.10.2 Additional annotation commands

Starting with texlive 2021 pdftex and luatex offer commands to interrupt a link. That can for example be used to exclude the header and footer from the link. We add here backend support for this.

```

974 <*drivers>
975 \cs_new_protected:Npn \__pdf_backend_link_off:{}
976 \cs_new_protected:Npn \__pdf_backend_link_on: {}
977 </drivers>
978 <*pdftex>
979 \cs_if_exist:NT \pdfrunninglinkoff
980 {
981     \cs_set_protected:Npn \__pdf_backend_link_off:
982     {
983         \pdfrunninglinkoff
984     }
985     \cs_set_protected:Npn \__pdf_backend_link_on:
986     {
987         \pdfrunninglinkon
988     }
989 }
990 </pdftex>
991 <*luatex>
992 \int_compare:nNnT {\tex_luatexversion:D } > {112}
993 {
994     \cs_set_protected:Npn \__pdf_backend_link_off:

```

```

995     {
996       \pdfextension linkstate 1
997     }
998   \cs_set_protected:Npn \__pdf_backend_link_on:
999     {
1000       \pdfextension linkstate 0
1001     }
1002   }
1003 </luatex>
1004 <*dvipdfmx | xdvipdfmx>
1005   \cs_set_protected:Npn \__pdf_backend_link_off:
1006     {
1007       \__pdf_backend:n { nolink }
1008     }
1009   \cs_set_protected:Npn \__pdf_backend_link_on:
1010     {
1011       \__pdf_backend:n { link }
1012     }
1013 </dvipdfmx | xdvipdfmx>

```

1.10.3 FormXObject / backend

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

```

\__pdf_backend_xform_use:n 1014 <*pdftex>
\__pdf_backend_xform_ref:n 1015 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
                             1016 % #1 name
                             1017 % #2 attributes
                             1018 % #3 resources
                             1019 % #4 content, not necessarily a box!
                             1020 {
                             1021   \hbox_set:Nn \l__pdf_backend_tmpa_box
                             1022     {
                             1023       \bool_set_true:N \l__pdf_backend_xform_bool
                             1024       \prop_gc_clear:c {\__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
                             1025       #4
                             1026     }
                             1027   %store the dimensions
                             1028   \tl_const:ce
                             1029     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
                             1030     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
                             1031   \tl_const:ce
                             1032     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
                             1033     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
                             1034   \tl_const:ce
                             1035     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
                             1036     { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
                             1037   %% do we need to test if #2 and #3 are empty??
                             1038   \tex_immediate:D \tex_pdfxform:D
                             1039     ~ attr ~ { #2 }

```

```

1040 %% which other resources should be default? Is an argument actually needed?
1041 ~ resources ~
1042 {
1043     #3
1044     \int_compare:nNnT
1045     { \prop_count:c { \__kernel_pdffdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1046     >
1047     { 0 }
1048     {
1049         /Properties~
1050         <<
1051         \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1052         >>
1053     }
1054
1055     \prop_if_empty:cF
1056     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1057     {
1058         /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1059     }
1060     \prop_if_empty:cF
1061     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1062     {
1063         /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1064     }
1065     \prop_if_empty:cF
1066     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1067     {
1068         /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1069     }
1070     \prop_if_empty:cF
1071     { \__kernel_pdffdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1072     {
1073         /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1074     }
1075 }
1076 \l__pdf_backend_tmpa_box
1077 \int_const:cn
1078 { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1079 { \tex_pdflastxform:D }
1080 }
1081
1082 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1083 {
1084     \tex_pdfrefxform:D
1085     \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1086     \scan_stop:
1087 }
1088
1089 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1090 {
1091     \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
1092 }
1093 </pdftex>

```

```

1094 <*luatex>
1095 %luatex
1096 %nearly identical but not completely ...
1097 \cs_new_protected:Npn \l__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1098 % #1 name
1099 % #2 attributes
1100 % #3 resources
1101 % #4 content, not necessarily a box!
1102 {
1103   \hbox_set:Nn \l__pdf_backend_tmpa_box
1104   {
1105     \bool_set_true:N \l__pdf_backend_xform_bool
1106     \prop_gc_clear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1107     #4
1108   }
1109   \tl_const:ce
1110   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1111   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1112   \tl_const:ce
1113   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1114   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1115   \tl_const:ce
1116   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1117   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1118   %% do we need to test if #2 and #3 are empty??
1119   \tex_immediate:D \tex_pdfxform:D
1120   ~ attr ~ { #2 }
1121   %% which resources should be default? Is an argument actually needed?
1122   ~ resources ~
1123   {
1124     #3
1125     \int_compare:nNnT
1126     { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } } }
1127     >
1128     { 0 }
1129     {
1130       /Properties~
1131       <<
1132       \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1133       >>
1134     }
1135     \prop_if_empty:cF
1136     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1137     {
1138       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1139     }
1140     \prop_if_empty:cF
1141     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1142     {
1143       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1144     }
1145     \prop_if_empty:cF
1146     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1147     {

```

```

1148         /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1149     }
1150     \prop_if_empty:cF
1151     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1152     {
1153         /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1154     }
1155 }
1156 \l__pdf_backend_tmpa_box
1157 \int_const:cn
1158 { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1159 { \tex_pdflastxform:D }
1160 }
1161
1162 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 %protected as with xelatex
1163 {
1164     \tex_pdfrefxform:D \int_use:c
1165     {
1166         c__pdf_backend_xform_ \tl_to_str:n {#1} _int
1167     }
1168     \scan_stop:
1169 }
1170
1171 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1172 { \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
1173
1174 </luatex>
1175 <*dvipdfmx | xdvipdfmx>
1176 % xetex
1177 % it needs a bit testing if it really works to set the box to 0 before the special ...
1178 % does it disturb viewing the xobject?
1179 % what happens with the resources (bdc)? (should work as they are specials too)
1180 % xetex requires that the special is in horizontal mode. This means it affects
1181 % typesetting. But we can no delay the whole form code to shipout
1182 % as the object reference and the size is often wanted on the current page.
1183 % so we need to allocate a box - but probably they won't be thousands xform
1184 % in a document so it shouldn't matter.
1185 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1186 % #1 name
1187 % #2 attributes
1188 % #3 resources
1189 % #4 content, not necessarily a box!
1190 {
1191     \int_gincr:N \g__pdf_backend_object_int
1192     \int_const:cn
1193     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1194     { \g__pdf_backend_object_int }
1195     \box_new:c { g__pdf_backend_xform_#1_box }
1196     \hbox_gset:cn { g__pdf_backend_xform_#1_box }
1197     {
1198         \bool_set_true:N \l__pdf_backend_xform_bool
1199         #4
1200     }
1201     \tl_const:ce

```

```

1202 { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1203 { \tex_the:D \box_wd:c { g__pdf_backend_xform_#1_box } }
1204 \tl_const:ce
1205 { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1206 { \tex_the:D \box_ht:c { g__pdf_backend_xform_#1_box } }
1207 \tl_const:ce
1208 { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1209 { \tex_the:D \box_dp:c { g__pdf_backend_xform_#1_box } }
1210 \box_set_dp:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1211 \box_set_ht:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1212 \box_set_wd:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1213 \hook_gput_next_code:nn {shipout/background}
1214 {
1215   \mode_leave_vertical: %needed, the xform disappears without it.
1216   \__pdf_backend:e
1217   {
1218     bxobj ~ \__pdf_backend_xform_ref:n { #1 }
1219     \c_space_tl width ~ \pdfxform_wd:n { #1 }
1220     \c_space_tl height ~ \pdfxform_ht:n { #1 }
1221     \c_space_tl depth ~ \pdfxform_dp:n { #1 }
1222   }
1223   \box_use_drop:c { g__pdf_backend_xform_#1_box }
1224   \__pdf_backend:e {put ~ @resources ~<<#3>> }
1225   \__pdf_backend:e
1226   {
1227     put~ @resources ~
1228     <<
1229       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1230     >>
1231   }
1232   \__pdf_backend:e
1233   {
1234     put~ @resources ~
1235     <<
1236       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1237     >>
1238   }
1239   \__pdf_backend:e
1240   {
1241     put~ @resources ~
1242     <<
1243       /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1244     >>
1245   }
1246   \__pdf_backend:e
1247   {
1248     put~ @resources ~
1249     <<
1250       /ColorSpace~
1251       \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1252     >>
1253   }
1254   \__pdf_backend:e {exobj ~<<#2>>}
1255 }

```

```

1256     }
1257
1258
1259
1260     \cs_new:Npn \__pdf_backend_xform_ref:n #1
1261     {
1262         @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1263     }
1264
1265     \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1266     {
1267         \hbox_set:Nn \l__pdf_backend_tmpa_box
1268         {
1269             \__pdf_backend:e
1270             {
1271                 uxobj~ \__pdf_backend_xform_ref:n { #1 }
1272             }
1273         }
1274         \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1275         \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1276         \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1277         \box_use_drop:N \l__pdf_backend_tmpa_box
1278     }
1279     </dvipdfmx|xdvipdfmx>
1280     <*dvisvgm>
1281     % unclear what it should do!!
1282     \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}
1283     \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
1284     \cs_new:Npn \__pdf_backend_xform_ref:n {}
1285     </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

```

1286     <*dvips>
1287     \tl_new:N \l__pdf_backend_xform_tmpwd_tl
1288     \tl_new:N \l__pdf_backend_xform_tmpdp_tl
1289     \tl_new:N \l__pdf_backend_xform_tmpht_tl
1290
1291     \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 % #1 name, #2 attribute, #4
1292     {
1293         \int_gincr:N \g__pdf_backend_object_int
1294         \int_const:cn
1295         { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1296         { \g__pdf_backend_object_int }
1297
1298         \hbox_set:Nn \l__pdf_backend_tmpa_box
1299         {
1300             \bool_set_true:N \l__pdf_backend_xform_bool
1301             \prop_gcload:c {\__kernel_pdffdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
1302             #4
1303         }
1304         %store the dimensions
1305         \tl_const:ce

```

```

1305     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1306     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1307 \tl_const:ce
1308     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1309     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1310 \tl_const:ce
1311     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1312     { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1313 %store content dimensions in DPI units (Dots) (code from issue 25)
1314 \tl_set:N\l__pdf_backend_xform_tmpwd_tl
1315 {
1316     \dim_to_decimal_in_sp:n{ \box_wd:N \l__pdf_backend_tmpa_box }~
1317     65536~div~72.27~div~DVImag~mul~Resolution~mul~
1318 }
1319 \tl_set:N\l__pdf_backend_xform_tmph_tl
1320 {
1321     \dim_to_decimal_in_sp:n{ \box_ht:N \l__pdf_backend_tmpa_box }~
1322     65536~div~72.27~div~DVImag~mul~VResolution~mul~
1323 }
1324 \tl_set:N\l__pdf_backend_xform_tmpdp_tl
1325 {
1326     \dim_to_decimal_in_sp:n{ \box_dp:N \l__pdf_backend_tmpa_box }~
1327     65536~div~72.27~div~DVImag~mul~VResolution~mul~
1328 }
1329 % mirror the box
1330 %\box_scale:Nnn \l__pdf_backend_tmpa_box {1} {-1}
1331 \hbox_set:Nn\l__pdf_backend_tmpl_box
1332 {
1333     \__kernel_backend_postscript:e
1334     {
1335         gsave~currentpoint~
1336         initclip~ % restore default clipping path (page device/whole page)
1337         clippath~pathbbox~newpath~pop~pop~
1338         \tl_use:N\l__pdf_backend_xform_tmpdp_tl~add~translate~
1339         mark~
1340         /_objdef~{ pdf.obj \int_use:N\g__pdf_backend_object_int }\c_space_tl~
1341         /BBox[
1342             0~
1343             \tl_use:N\l__pdf_backend_xform_tmph_tl~
1344             \tl_use:N\l__pdf_backend_xform_tmpwd_tl~
1345             \tl_use:N\l__pdf_backend_xform_tmpdp_tl~
1346             neg
1347         ]
1348         \str_if_eq:eeF{#1}{}
1349         {
1350             product~(Distiller)~search~{pop~pop~pop~#2}{pop}ifelse~
1351         }
1352         /BP~pdfmark~1~-1~scale~neg~exch~neg~exch~translate
1353     }
1354     \box_use_drop:N\l__pdf_backend_tmpa_box
1355     \__kernel_backend_postscript:n
1356     {
1357         mark ~ /EP~pdfmark ~ grestore
1358     }

```

```

1359     \str_if_eq:eeF{#1}{}
1360     {
1361         \__kernel_backend_postscript:e
1362         {
1363             product~(Ghostscript)~search~
1364             {
1365                 pop~pop~pop~
1366                 mark~
1367                 { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1368                 ~<<#2>>~/PUT~pdfmark
1369             }{pop}ifelse
1370         }
1371     }
1372 }
1373 \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1374 \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1375 \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1376 \hook_gput_code:nnn {begindocument/end}{pdfxform}
1377 {
1378     \mode_leave_vertical:
1379     \box_use:N\l__pdf_backend_tmpb_box
1380 }
1381 }
1382
1383
1384 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1385 {
1386     \hbox_set:Nn \l__pdf_backend_tmpa_box
1387     {
1388         \__kernel_backend_postscript:e
1389         {
1390             gsave~currentpoint~translate~1~-1~scale~
1391             mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }~
1392             /SP~pdfmark ~ grestore
1393         }
1394     }
1395     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1396     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1397     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1398     \box_use_drop:N \l__pdf_backend_tmpa_box
1399 }
1400 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1401 {
1402     { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1403 }
1404
1405 </dvips>
1406 <*drivers>
1407 %% all
1408 \prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
1409 {
1410     \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1411     { \prg_return_true: }
1412     { \prg_return_false: }

```

```

1413 }
1414 \prg_new_eq_conditional:Nn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
1415 { TF , T , F , p }
1416 </drivers>

```

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

1.11 Structure Destinations

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

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

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

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

`\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
1417 <*drivers>
1418 \tl_new:N \l_pdf_current_structure_destination_tl
1419 </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:nnnn -> \__pdf_backend_structure_destination:nnnn
\__pdf_backend_link_begin_goto:nnw -> \__pdf_backend_link_begin_structure_goto:nnw

```

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:

```

1420 <*drivers>
1421 \cs_new_protected:Npn \pdf_activate_structure_destination:
1422 {
1423   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_structure_destination:nn
1424   \cs_gset_eq:NN \__pdf_backend_destination:nnnn \__pdf_backend_structure_destination:nnnn
1425   \cs_gset_eq:NN \__pdf_backend_link_begin_goto:nnw \__pdf_backend_link_begin_structure_goto:nnw
1426 }
1427 </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.

```

1428 <*drivers>
1429 \cs_set_eq:NN \__pdf_backend_structure_destination:nn \__pdf_backend_destination:nn
1430 \cs_set_eq:NN \__pdf_backend_structure_destination:nnnn \__pdf_backend_destination:nnnn
1431 \cs_set_eq:NN \__pdf_backend_link_begin_structure_goto:nnw \__pdf_backend_link_begin_goto:nnw
1432 </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:

```

1433 <*xdvipdfmx | dvipdfmx>
1434 \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1435 {
1436   \__pdf_backend:e
1437   {
1438     dest ~ ( \exp_not:n {#1} )
1439     [
1440       @thispage
1441       \str_case:nnF {#2}
1442       {
1443         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1444         { fit } { /Fit }
1445         { fitb } { /FitB }
1446         { fitbh } { /FitBH }
1447         { fitbv } { /FitBV ~ @xpos }
1448         { fith } { /FitH ~ @ypos }
1449         { fitv } { /FitV ~ @xpos }
1450         { fitr } { /Fit }
1451       }
1452       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1453     ]
1454   }

```

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.

```

1455   \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1456   {
1457     \__pdf_backend:e

```

```

1458 {
1459   obj ~ @pdf.SDest.\exp_not:n{#1}
1460   [
1461     \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1462     \str_case:nnF {#2}
1463     {
1464       { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1465       { fit } { /Fit }
1466       { fitb } { /FitB }
1467       { fitbh } { /FitBH }
1468       { fitbv } { /FitBV ~ @xpos }
1469       { fith } { /FitH ~ @ypos }
1470       { fitv } { /FitV ~ @xpos }
1471       { fitr } { /Fit }
1472     }
1473     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1474   ]
1475 }
1476 }
1477 }

```

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

```

1478 \cs_new_protected:Npn \__pdf_backend_structure_destination_aux:nnnn #1#2#3#4
1479 {
1480   \vbox_to_zero:n
1481   {
1482     \__kernel_kern:n {#4}
1483     \hbox:n
1484     {
1485       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1486       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1487     }
1488     \tex_vss:D
1489   }
1490   \__kernel_kern:n {#1}
1491   \vbox_to_zero:n
1492   {
1493     \__kernel_kern:n { -#3 }
1494     \hbox:n
1495     {
1496       \__pdf_backend:n
1497       {
1498         dest ~ (#2)
1499         [
1500           @thispage
1501           /FitR ~
1502           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1503           @xpos ~ @ypos
1504         ]
1505       }
1506     }
1507   }

```

Here we add the structure destination to the same box

```

1506   \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1507   {

```

```

1508         \_pdf_backend:e
1509         {
1510             obj ~ @pdf.SDest.\exp_not:n{#2}
1511             [
1512                 \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_
1513                 /FitR ~
1514                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1515                 @xpos ~ @ypos
1516             ]
1517         }
1518     }
1519 }
1520 \tex_vss:D
1521 }
1522 \_kernel_kern:n { -#1 }
1523 }

```

And now we redefine the destination command:

```

1524 \cs_set_protected:Npn \_pdf_backend_structure_destination:nnnn #1#2#3#4
1525 {
1526     \exp_args:Ne \_pdf_backend_structure_destination_aux:nnnn
1527     { \dim_eval:n {#2} } {#1} {#3} {#4}
1528 }

```

At last the goto link.

```

1529 \cs_set_protected:Npn \_pdf_backend_link_begin_structure_goto:nnw #1#2
1530 {
1531     \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD~@pdf.SDest.
1532 }
1533 </xdvipdfmx | dvipdfmx>

```

(End of definition for _pdf_backend_structure_destination:nn.)

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

```

1534 <*pdftex>
1535 \bool_lazy_and:nnT
1536 { \int_compare_p:nNn {\tex_pdftexversion:D} > {139} }
1537 { \int_compare_p:nNn {\tex_pdftexrevision:D} > {23} }
1538 {
1539     \cs_set_protected:Npn \_pdf_backend_structure_destination:nn #1#2
1540     {
1541         \tex_pdfdest:D
1542         name {#1}
1543         \str_case:nnF {#2}
1544         {
1545             { xyz } { xyz }
1546             { fit } { fit }
1547             { fitb } { fitb }
1548             { fitbh } { fitbh }
1549             { fitbv } { fitbv }
1550             { fith } { fith }
1551             { fitv } { fitv }
1552             { fitr } { fitr }
1553         }
1554         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1555         \scan_stop:

```

```

1556 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1557 {
1558   \tex_pdfdest:D
1559   struct~
1560   \int_use:c
1561   { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1562   name {#1}
1563   \str_case:nnF {#2}
1564   {
1565     { xyz } { xyz }
1566     { fit } { fit }
1567     { fitb } { fitb }
1568     { fitbh } { fitbh }
1569     { fitbv } { fitbv }
1570     { fith } { fith }
1571     { fitv } { fitv }
1572     { fitr } { fitr }
1573   }
1574   { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1575   \scan_stop:
1576 }
1577 }
1578 \cs_set_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
1579 {
1580   \tex_pdfdest:D
1581   name {#1}
1582   fitr ~
1583   width \dim_eval:n {#2} ~
1584   height \dim_eval:n {#3} ~
1585   depth \dim_eval:n {#4} \scan_stop:
1586 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1587 {
1588   \tex_pdfdest:D
1589   struct~
1590   \int_use:c
1591   { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_
1592   name {#1}
1593   fitr ~
1594   width \dim_eval:n {#2} ~
1595   height \dim_eval:n {#3} ~
1596   depth \dim_eval:n {#4} \scan_stop:
1597 }
1598 }
1599 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1600 {
1601   \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1602 }
1603 }
1604 </pdftex>

```

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

```

1605 <*luatex>
1606 \int_compare:nNnT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1607 {
1608   \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2

```

```

1609 {
1610   \tex_pdfextension:D dest
1611     name {#1}
1612     \str_case:nnF {#2}
1613     {
1614       { xyz } { xyz }
1615       { fit } { fit }
1616       { fitb } { fitb }
1617       { fitbh } { fitbh }
1618       { fitbv } { fitbv }
1619       { fith } { fith }
1620       { fitv } { fitv }
1621       { fitr } { fitr }
1622     }
1623     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1624     \scan_stop:
1625   \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1626   {
1627     \tex_pdfextension:D dest
1628     struct~
1629     \int_use:c
1630     { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1631     name {#1}
1632     \str_case:nnF {#2}
1633     {
1634       { xyz } { xyz }
1635       { fit } { fit }
1636       { fitb } { fitb }
1637       { fitbh } { fitbh }
1638       { fitbv } { fitbv }
1639       { fith } { fith }
1640       { fitv } { fitv }
1641       { fitr } { fitr }
1642     }
1643     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1644     \scan_stop:
1645   }
1646 }
1647 \cs_set_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
1648 {
1649   \tex_pdfextension:D dest
1650   name {#1}
1651   fitr ~
1652   width \dim_eval:n {#2} ~
1653   height \dim_eval:n {#3} ~
1654   depth \dim_eval:n {#4} \scan_stop:
1655   \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1656   {
1657     \tex_pdfextension:D dest
1658     struct~
1659     \int_use:c
1660     { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_
1661     name {#1}
1662     fitr ~

```

```

1663         width \dim_eval:n {#2} ~
1664         height \dim_eval:n {#3} ~
1665         depth \dim_eval:n {#4} \scan_stop:
1666     }
1667 }
1668 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1669 {
1670     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1671 }
1672 }
1673 </luatex>

```

1.12 Settings for regression tests

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

```

1674 <*drivers>
1675 \cs_new_protected:Npn \__pdf_backend_set_regression_data:
1676 {
1677     \sys_gset_rand_seed:n{1000}
1678     \pdfmanagement_add:nnn{Info}{Creator}{(TeX)}
1679 </drivers>
1680 <*dvips>
1681     \AddToHook{begindocument}{\pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX+dvips)}}
1682     \__kernel_backend_literal:e{!~<</DocumentUUID~(DocumentUUID)>>~setpagedevice}
1683     \__kernel_backend_literal:e{!~<</InstanceUUID~(InstanceUUID)>>~setpagedevice}
1684     \str_if_exist:NTF\c_sys_timestamp_str
1685     {
1686         \pdfmanagement_add:nne{Info}{CreationDate}{(\c_sys_timestamp_str)}
1687         \pdfmanagement_add:nne{Info}{ModDate}{(\c_sys_timestamp_str)}
1688     }
1689     {
1690         \pdfmanagement_add:nnn{Info}{CreationDate}{(D:20010101205959-00'00')}
1691         \pdfmanagement_add:nnn{Info}{ModDate}{(D:20010101205959-00'00')}
1692     }
1693 </dvips>
1694 <*dvipdfmx>
1695     \pdfmanagement_add:nnn{Info}{Producer}{(dvipdfmx)}
1696     \__kernel_backend_literal:e
1697     {pdf:trailerid [~
1698     <00112233445566778899aabbccddeeff>~
1699     <00112233445566778899aabbccddeeff>~
1700     ]}
1701 </dvipdfmx>
1702 <*xdvipdfmx>
1703     \pdfmanagement_add:nnn{Info}{Producer}{(xetex)}
1704     \__kernel_backend_literal:e
1705     {pdf:trailerid [~
1706     <00112233445566778899aabbccddeeff>~
1707     <00112233445566778899aabbccddeeff>~
1708     ]}
1709 </xdvipdfmx>

```

```

1710 <*pdftex>
1711   \pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX)}
1712   \tex_pdfsuppressptexinfo:D 7 \scan_stop:
1713   \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
1714 </pdftex>
1715 <*luatex>
1716   \pdfmanagement_add:nnn{Info}{Producer}{(LuaTeX)}
1717   \tex_pdfvariable:D suppressoptionalinfo 7\relax
1718   \tex_pdfvariable:D trailerid
1719   {[~
1720     <2350CAD05F8A7AF0AA4058486855344F>~
1721     <2350CAD05F8A7AF0AA4058486855344F>~
1722   ]}
1723 </luatex>
1724 <*drivers>
1725   \str_if_exist:NF\c_sys_timestamp_str
1726   {
1727     \pdfmanagement_add:nnn{Info}{CreationDate}{(D:20010101205959-00'00')}
1728     \pdfmanagement_add:nnn{Info}{ModDate}{(D:20010101205959-00'00')}
1729     \AddToDocumentProperties[document]{creationdate}{D:20010101205959-00'00'}
1730     \AddToDocumentProperties[document]{moddate}{D:20010101205959-00'00'}
1731     \AddToDocumentProperties[hyperref]{pdfmetadate}{D:20010101205959-00'00'}
1732     \AddToDocumentProperties[hyperref]{pdfdate}{D:20010101205959-00'00'}
1733   }
1734   \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-8c19-6237d832f}
1735   \AddToDocumentProperties[hyperref]{pdfproducer}{\c_sys_engine_exec_str-NN.NN.NN}
1736 }
1737 </drivers>

```

1.13 Uncompressed metadata object stream

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

```

1738 <*luatex>
1739 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1740 {
1741   \tex_immediate:D \tex_pdfextension:D obj ~uncompressed~
1742   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1743 }
1744 </luatex>
1745 <*pdftex>
1746 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1747 {
1748   \group_begin:
1749   \tex_pdfcompresslevel:D 0 \scan_stop:
1750   \tex_immediate:D \tex_pdfobj:D
1751   \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}

```

```

1752     \group_end:
1753   }
1754   </pdftex>
1755   <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1756   \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1757   {
1758     \pdf_object_unnamed_write:nn {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
1759   }
1760   </xdvipdfmx | dvipdfmx | dvips | dvisvgm>

```

1.14 Suppressing deprecated PDF features

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

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

```

1761   <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1762   \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 {} % #1 number
1763   </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1764   <*pdftex>
1765   \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 % #1 number
1766   {
1767     \tex_pdfomitcharset:D = #1 \scan_stop:
1768   }
1769   </pdftex>
1770   <*luatex>
1771   \cs_new_protected:Npn \__pdf_backend_omit_charset:n #1 % #1 number
1772   {
1773     \tex_pdfvariable:D omitcharset = #1 \scan_stop:
1774   }
1775   </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.

```

1776   <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1777   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {} % #1 number
1778   </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1779   <*pdftex>
1780   \bool_lazy_and:nnTF
1781   { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1782   { \int_compare_p:nNn {\tex_pdftexrevision:D } > {24} }
1783   {
1784     \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 % #1 number
1785     {
1786       \pdfomitinfodict = #1 \scan_stop:
1787     }
1788   }
1789   {
1790     \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {} % #1 number
1791   }
1792   }

```

```

1793 </pdfTeX>
1794 <*luatex>
1795 \int_compare:nNnTF {\directlua{tex.print(status.list()["development_id"])} } > {7560}
1796 {
1797   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {%#1 number
1798   {
1799     \tex_pdfvariable:D omitinfodict = #1 \scan_stop:
1800   }
1801 }
1802 {
1803   \cs_new_protected:Npn \__pdf_backend_omit_info:n #1 {} {%#1 number
1804 }
1805 </luatex>

```

(End of definition for __pdf_backend_omit_info:n.)

1.15 lua code for luatex

```

1806 <*lua>
1807 ltx= ltx or {}
1808 ltx.__pdf = ltx.__pdf or {}
1809 ltx.__pdf.Page = ltx.__pdf.Page or {}
1810 ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
1811 ltx.__pdf.Page.Resources = ltx.__pdf.Page.Resources or {}
1812 ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
1813 ltx.__pdf.Page.Resources.List={"ExtGState","ColorSpace","Pattern","Shading"}
1814 ltx.__pdf.object = ltx.__pdf.object or {}
1815
1816 ltx.pdf= ltx.pdf or {} -- for "public" functions
1817
1818 local __pdf = ltx.__pdf
1819 local pdf = pdf
1820
1821 local function __pdf_backend_Page_gput (name,value)
1822   __pdf.Page.dflt[name]=value
1823 end
1824
1825 local function __pdf_backend_Page_gremove (name)
1826   __pdf.Page.dflt[name]=nil
1827 end
1828
1829 local function __pdf_backend_Page_gclear ()
1830   __pdf.Page.dflt={}
1831 end
1832
1833 local function __pdf_backend_ThisPage_gput (page,name,value)
1834   __pdf.Page[page] = __pdf.Page[page] or {}
1835   __pdf.Page[page][name]=value
1836 end
1837
1838 local function __pdf_backend_ThisPage_gpush (page)
1839   local token=""
1840   local t = {}
1841   local tkeys= {}

```

```

1842 for name,value in pairs(__pdf.Page.dflt) do
1843     t[name]=value
1844 end
1845 if __pdf.Page[page] then
1846     for name,value in pairs(__pdf.Page[page]) do
1847         t[name] = value
1848     end
1849 end
1850 -- sort the table to get reliable test files.
1851 for name,value in pairs(t) do
1852     table.insert(tkeys,name)
1853 end
1854 table.sort(tkeys)
1855 for _,name in ipairs(tkeys) do
1856     token = token .. "/"..name.." "..t[name]
1857 end
1858 return token
1859 end
1860
1861 function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly_
1862 __pdf_backend_ThisPage_gput (page,name,value)
1863 end
1864
1865 function ltx.__pdf.backend_ThisPage_gpush (page)
1866     pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
1867 end
1868
1869 function ltx.__pdf.backend_Page_gput (name,value)
1870     __pdf_backend_Page_gput (name,value)
1871 end
1872
1873 function ltx.__pdf.backend_Page_gremove (name)
1874     __pdf_backend_Page_gremove (name)
1875 end
1876
1877 function ltx.__pdf.backend_Page_gclear ()
1878     __pdf_backend_Page_gclear ()
1879 end
1880
1881
1882 local Properties = ltx.__pdf.Page.Resources.Properties
1883 local ResourceList= ltx.__pdf.Page.Resources.List
1884 local function __pdf_backend_PageResources_gpush (page)
1885     local token=""
1886     if Properties[page] then
1887         -- we sort the table, so that the pdf test works
1888         local t = {}
1889         for name,value in pairs (Properties[page]) do
1890             table.insert (t,name)
1891         end
1892         table.sort (t)
1893         for _,name in ipairs(t) do
1894             token = token .. "/"..name.." ".. Properties[page][name]
1895         end

```

```

1896 token = "/Properties <<\"..token..\">>"
1897 end
1898 for i,name in ipairs(ResourceList) do
1899   if ltx._pdf.Page.Resources[name] then
1900     token = token .. "/" .. name .. " " .. ltx.pdf.object_ref("__pdf/Page/Resources/" .. name)
1901   end
1902 end
1903 return token
1904 end
1905
1906 -- the function is public, as I probably need it in tagpdf too ...
1907 function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
1908   Properties[page] = Properties[page] or {}
1909   Properties[page][name]=value
1910   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
1911 end
1912
1913 function ltx.pdf.Page_Resources_gpush(page)
1914   pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
1915 end
1916
1917 function ltx.pdf.object_ref (objname)
1918   if ltx._pdf.object[objname] then
1919     local ref= ltx._pdf.object[objname]
1920     return ref
1921   else
1922     return "false"
1923   end
1924 end
1925 </lua>

```

Index

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

A		<code>\box_new:N</code> 106, 107, 1195
<code>\AddToDocumentProperties</code>		<code>\box_scale:Nnn</code> 1330
.. 1729, 1730, 1731, 1732, 1734, 1735		<code>\box_set_dp:Nn</code> 1210, 1276, 1373, 1397
<code>\AddToHook</code>	1681	<code>\box_set_ht:Nn</code> 1211, 1275, 1374, 1396
B		<code>\box_set_wd:Nn</code> 1212, 1274, 1375, 1395
bool commands:		<code>\box_use:N</code> 1379
<code>\bool_if:NTF</code> 39, 556, 589, 670, 676,		<code>\box_use_drop:N</code> 1223, 1277, 1354, 1398
712, 736, 771, 777, 803, 833, 873, 878		<code>\box_wd:N</code> . 1030, 1111, 1203, 1306, 1316
<code>\bool_lazy_and:nnTF</code>	1535, 1780	
<code>\bool_new:N</code>	542	
<code>\bool_set_true:N</code> 1023, 1105, 1198, 1299		
box commands:		C
<code>\box_dp:N</code> . 1036, 1117, 1209, 1312, 1326		clist commands:
<code>\box_ht:N</code> . 1033, 1114, 1206, 1309, 1321		<code>\clist_const:Nn</code> 438
		<code>\clist_map_function:NN</code> 916
		<code>\clist_map_inline:Nn</code> 447, 481, 497, 692

cs commands:

`\cs_generate_variant:Nn`
 28, 31, 32, 35, 36, 97, 98, 435
`\cs_gset_eq:NN` 671, 672,
 772, 773, 874, 875, 1423, 1424, 1425
`\cs_if_exist:NTF` 69, 450, 979
`\cs_new:Npn` 77, 91, 118, 124,
 266, 892, 1089, 1171, 1260, 1284, 1400
`\cs_new_protected:Npn`
 41, 45, 55, 71, 84, 168,
 177, 193, 199, 205, 212, 219, 228,
 248, 271, 281, 295, 307, 324, 335,
 342, 349, 358, 367, 374, 381, 388,
 397, 406, 414, 417, 423, 428, 431,
 462, 473, 479, 505, 509, 521, 524,
 525, 529, 532, 533, 537, 558, 581,
 602, 690, 791, 901, 925, 932, 939,
 948, 952, 955, 958, 970, 975, 976,
 1015, 1082, 1097, 1162, 1185, 1265,
 1282, 1283, 1290, 1384, 1421, 1478,
 1675, 1739, 1746, 1756, 1762, 1765,
 1771, 1777, 1784, 1790, 1797, 1803
`\cs_new_protected:Npx` 187
`\cs_set_eq:NN`
 682, 783, 884, 1429, 1430, 1431
`\cs_set_protected:Npn` 551, 565, 569,
 573, 577, 587, 591, 594, 596, 598,
 600, 613, 632, 651, 657, 663, 668,
 678, 684, 707, 731, 755, 759, 764,
 769, 779, 786, 796, 826, 857, 861,
 866, 871, 880, 887, 981, 985, 994,
 998, 1005, 1009, 1434, 1524, 1529,
 1539, 1578, 1599, 1608, 1647, 1668

D

dim commands:

`\dim_eval:n` 1527,
 1583, 1584, 1585, 1594, 1595, 1596,
 1652, 1653, 1654, 1663, 1664, 1665
`\dim_to_decimal_in_sp:n`
 1316, 1321, 1326
`\c_zero_dim`
 1210, 1211, 1212, 1373, 1374, 1375
`\directlua` 115, 1606, 1795

E

exp commands:

`\exp_args:Ne` 720, 744, 1455,
 1461, 1506, 1512, 1526, 1556, 1561,
 1586, 1591, 1625, 1630, 1655, 1660
`\exp_args:NNe` 903
`\exp_not:n`
 637, 735, 830, 1438, 1459, 1510

F

fp commands:

`\fp_eval:n`
 1452, 1473, 1554, 1574, 1623, 1643

G

group commands:

`\group_begin:` 1748
`\group_end:` 1752

H

hbox commands:

`\hbox:n` 1483, 1494
`\hbox_gset:Nn` 1196
`\hbox_set:Nn`
 1021, 1103, 1267, 1297, 1331, 1386

hook commands:

`\hook_gput_code:nnn` .. 160, 500, 1376
`\hook_gput_next_code:nn` 1213
`\hook_gset_rule:nnnn` 494, 495

I

int commands:

`\int_compare:nNnTF`
 992, 1044, 1125, 1606, 1795
`\int_compare_p:nNn`
 1536, 1537, 1781, 1782
`\int_const:Nn` . 1077, 1157, 1192, 1293
`\int_gincr:N` 231, 615, 634,
 709, 733, 798, 802, 828, 832, 1191, 1292
`\int_if_exist:NTF` 1410
`\int_new:N` 110, 111, 112
`\int_use:N` 232, 235,
 618, 626, 637, 645, 711, 716, 725,
 735, 740, 749, 800, 807, 811, 814,
 822, 830, 837, 841, 844, 852, 1085,
 1091, 1164, 1172, 1262, 1340, 1367,
 1391, 1402, 1560, 1590, 1629, 1659

K

kernel internal commands:

`__kernel_backend_literal:n`
 31, 101, 616, 620, 635, 639, 653,
 665, 686, 696, 1682, 1683, 1696, 1704
`__kernel_backend_literal_page:n`
 28, 710, 734,
 757, 766, 788, 799, 829, 859, 868, 889
`__kernel_backend_postscript:n` ..
 35, 1333, 1355, 1361, 1388
`__kernel_backend_shipout_-`
 literal:n 39, 41, 560, 680
`__kernel_backend_shipout_-`
 literal_page:n ... 55, 55, 781, 882
`__kernel_backend_shipout_-`
 literal_pdf:n 45, 45

`__kernel_kern:n` 1482, 1490, 1493, 1522
`__kernel_pdf_name_from_unicode_-`
`e:n` 118, 124
`__kernel_pdfdict_name:n`
`.....` 250, 251, 253,
`484, 512, 694, 895, 906, 911, 1024,`
`1045, 1056, 1061, 1066, 1071, 1106,`
`1126, 1136, 1141, 1146, 1151, 1300`
`\g__kernel_pdfmanagement_end_-`
`run_code_tl` 133, 140, 147
`\g__kernel_pdfmanagement_-`
`thispage_shipout_code_tl` 156, 162

L

l^{at}elua commands:

`\latelua:` 225, 304, 355, 394

M

mode commands:

`\mode_leave_vertical:` ... 1215, 1378

P

pdf commands:

`\pdf_activate_structure_destination:`
`.....` 1420, 1421
`\l_pdf_current_structure_-`
`destination_tl` 1417,
`1455, 1461, 1506, 1512, 1556, 1561,`
`1586, 1591, 1625, 1630, 1655, 1660`
`\pdf_object_if_exist:nTF`
`..` 1455, 1506, 1556, 1586, 1625, 1655
`\pdf_object_new:n` 449, 499
`\pdf_object_ref:n` 897, 1058, 1063,
`1068, 1073, 1138, 1143, 1148, 1153,`
`1229, 1236, 1243, 1251, 1461, 1512`
`\pdf_object_ref_last:` . 928, 935, 942
`\pdf_object_unnamed_write:nn` ...
`...` 659, 761, 863, 927, 934, 941, 1758
`\pdf_object_write` 514
`\pdf_object_write:nnn` 486, 503

pdf internal commands:

`__pdf_backend:n` 32, 195, 507,
`515, 942, 1007, 1011, 1216, 1224,`
`1225, 1232, 1239, 1246, 1254, 1269,`
`1436, 1457, 1485, 1486, 1496, 1508`
`__pdf_backend_bdc:nn` 13,
`539, 551, 587, 668, 671, 672, 673,`
`769, 772, 773, 774, 871, 874, 875, 876`
`__pdf_backend_bdc_contobj:nn` ...
`.....` 657, 671, 759, 772, 861, 874
`__pdf_backend_bdc_contstream:nn`
`.....` 663, 672, 764, 773, 866, 875
`__pdf_backend_bdc_shipout:nn` ...
`.....` 558, 682, 783, 884

`__pdf_backend_bdc_shipout_-`
`contstream:nn`
`.....` 678, 682, 779, 783, 880, 884
`__pdf_backend_bdcobject:n`
`.....` 13, 539,
`569, 596, 632, 660, 731, 762, 826, 864`
`__pdf_backend_bdcobject:nn`
`.....` 13, 539, 565, 594, 613, 707, 796
`__pdf_backend_bmc:n`
`.....` 13, 539, 577, 600, 651, 755, 857
`__pdf_backend_catalog_gput:nn` .. 21
`__pdf_backend_destination:nn` ...
`.....` 1423, 1429
`__pdf_backend_destination:nnnn` .
`.....` 1424, 1430, 1578, 1647
`__pdf_backend_emc:`
`.....` 13, 539, 573, 598, 684, 786, 887
`__pdf_backend_link_begin:n` .. 1531
`__pdf_backend_link_begin:nnnw` ..
`.....` 1601, 1670
`__pdf_backend_link_begin_-`
`goto:nnw` 1425, 1431
`__pdf_backend_link_begin_-`
`structure_goto:nnw`
`.....` 1425, 1431, 1529, 1599, 1668
`__pdf_backend_link_off:`
`.....` 975, 981, 994, 1005
`__pdf_backend_link_on:`
`.....` 976, 985, 998, 1009
`__pdf_backend_luastring:n`
`.....` 181, 266, 275, 287, 288, 299, 314, 315
`__pdf_backend_metadata_stream:n`
`.....` 1739, 1746, 1756
`\g__pdf_backend_name_int`
`.....` 109, 615, 618, 626,
`634, 637, 645, 709, 711, 716, 725,`
`733, 735, 740, 749, 798, 800, 828, 830`
`__pdf_backend_Names_gpush:nn` ...
`.....` 925, 932, 939, 948, 952
`__pdf_backend_NamesEmbeddedFiles_-`
`add:nn` 954, 955, 958, 970
`\g__pdf_backend_object_int`
`.....` 1191, 1194, 1292, 1295, 1340
`__pdf_backend_object_last:`
`.....` 571, 646, 741, 750, 838, 853
`__pdf_backend_object_ref:n` 456,
`517, 567, 627, 699, 717, 726, 808, 823`
`__pdf_backend_object_write:nn` ..
`.....` 1742, 1751
`__pdf_backend_omit_charset:n` ...
`.....` 1761, 1762, 1765, 1771
`__pdf_backend_omit_info:n`
`..` 1776, 1777, 1784, 1790, 1797, 1803

_pdf_backend_Page_gput:nn 1274, 1275, 1276, 1277, 1297, 1306,
 6, 202, 212, 281, 342, 381, 417 1309, 1312, 1316, 1321, 1326, 1330,
 _pdf_backend_Page_gremove:n 1354, 1386, 1395, 1396, 1397, 1398
 6, 202, 219, 295, 349, 388, 423
 \g_pdf_backend_page_int 109
 _pdf_backend_Page_primitive:n
 6, 202, 205, 258, 271, 736, 803, 833, 1023, 1105, 1198, 1299
 335, 360, 369, 374, 399, 408, 414, 435
 _pdf_backend_PageResources:n
 505, 524, 532
 \c_pdf_backend_PageResources_-clist . . 437, 447, 481, 497, 692, 917
 _pdf_backend_PageResources_-gpush:n
 14, 539, 581, 602, 690, 791, 901
 _pdf_backend_PageResources_-gpush_aux:n 892, 918
 _pdf_backend_PageResources_-gput:nnn 446, 462, 473, 509, 525, 533
 _pdf_backend_PageResources_-obj_gpush: . 446, 479, 521, 529, 537
 _pdf_backend_Pages_primitive:n
 167, 168, 177, 187, 193, 199
 _pdf_backend_pdfmark:n
 36, 553, 567, 571, 575, 579, 960
 _pdf_backend_record_abspage:n
 71, 84, 97, 232, 811, 841
 _pdf_backend_ref_abspage:n
 77, 91, 98, 235, 814, 844
 _pdf_backend_ref_label:nn 85
 _pdf_backend_ref_value:nn 92
 \g_pdf_backend_resourceid_int
 109, 231, 232, 235, 802, 807, 811, 814, 822, 832, 837, 841, 844, 852
 _pdf_backend_set_regression_-data: 1675
 _pdf_backend_shipout_bdc:nn
 13, 539, 591
 _pdf_backend_structure_-destination:nn
 1423, 1429, 1433, 1434, 1539, 1608
 _pdf_backend_structure_-destination:nnnn 1424, 1430, 1524
 _pdf_backend_structure_-destination_aux:nnnn . . 1478, 1526
 _pdf_backend_ThisPage_gpush:n
 6, 202, 248, 324, 367, 406, 431
 _pdf_backend_ThisPage_gput:nn
 6, 202, 228, 307, 358, 397, 428
 \g_pdf_backend_thispage_-shipout_tl 6
 \l_pdf_backend_tmpa_box
 103, 1021, 1030, 1033, 1036, 1076, 1103, 1111, 1114, 1117, 1156, 1267,
 1274, 1275, 1276, 1277, 1297, 1306,
 1309, 1312, 1316, 1321, 1326, 1330,
 1354, 1386, 1395, 1396, 1397, 1398
 \l_pdf_backend_tmpb_box
 107, 1331, 1373, 1374, 1375, 1379
 \l_pdf_backend_xform_bool
 542, 712, 736, 803, 833, 1023, 1105, 1198, 1299
 _pdf_backend_xform_if_exist:n
 1408, 1414
 _pdf_backend_xform_new:nnnn
 1014, 1015, 1097, 1185, 1282, 1290
 _pdf_backend_xform_ref:n
 1014, 1089, 1171, 1218, 1260, 1271, 1284, 1400
 \l_pdf_backend_xform_tmpdp_tl
 1288, 1324, 1338, 1345
 \l_pdf_backend_xform_tmphl_tl
 1289, 1319, 1343
 \l_pdf_backend_xform_tmpwd_tl
 1287, 1314, 1344
 _pdf_backend_xform_use:n
 1014, 1082, 1162, 1265, 1283, 1384
 \g_pdf_tmpa_prop . . 103, 250, 255, 260
 \l_pdf_tmpa_tl
 103, 233, 237, 239, 242, 812, 816, 818, 821, 842, 846, 848, 851, 854
 pdfdict commands:
 \pdfdict_gput:nnn
 214, 242, 344, 383, 419, 464, 475, 527, 535, 714, 738, 805, 820, 835, 850
 \pdfdict_gremove:nn 221, 351, 390, 425
 \pdfdict_if_exist:nTF . . 237, 816, 846
 \pdfdict_item:nn 260, 897, 912
 \pdfdict_new:n 239, 818, 848
 \pdfdict_show:n 854
 \pdfdict_use:n 370, 409, 488, 1051, 1132
 \pdfextension 996, 1000
 \pdfliteral 2
 pdfmanagement commands:
 \pdfmanagement_add:nnn
 1678, 1681, 1686, 1687, 1690, 1691, 1695, 1703, 1711, 1716, 1727, 1728
 pdfmanagement internal commands:
 \g_pdfmanagement_active_bool
 670, 771, 873
 \l_pdfmanagement_delayed_-shipout_bool
 39, 556, 589, 676, 777, 878
 \pdfnames 21
 \pdfomitinfodict 1786
 \pdfpageref 4
 \pdfrunninglinkoff 979, 983
 \pdfrunninglinkon 987

<code>\tl_const:Nn</code>	1029, 1032, 1035, 1078, 1085,
1028, 1031, 1034, 1109, 1112, 1115,	1091, 1110, 1113, 1116, 1158, 1166,	
1201, 1204, 1207, 1304, 1307, 1310	1172, 1193, 1202, 1205, 1208, 1262,	
<code>\tl_gput_right:Nn</code>	1294, 1305, 1308, 1311, 1367, 1391,	
131, 138, 145	1402, 1410, 1561, 1591, 1630, 1660	
<code>\tl_if_exist:NTF</code>	<code>\tl_use:N</code>	1338, 1343, 1344, 1345
151		
<code>\tl_new:N</code> ..		
105, 1287, 1288, 1289, 1418		
<code>\tl_set:Nn</code>		V
..... 233, 812, 842, 1314, 1319, 1324	vbox commands:	
<code>\tl_to_str:n</code>	<code>\vbox_to_zero:n</code>	1480, 1491