

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

The L^AT_EX Project*

Version 0.95t, released 2022-11-03

1 l3backend-testphase Implementation

```
1 <drivers>\ProvidesExplFile
2 <*dvipdfmx>
3   {l3backend-testphase-dvipdfmx.def}{2022-11-03}{}
4   {LaTeX-PDF-management-testphase-bundle-backend-support: dvipdfmx}
5 </dvipdfmx>
6 <*dvips>
7   {l3backend-testphase-dvips.def}{2022-11-03}{}
8   {LaTeX-PDF-management-testphase-bundle-backend-support: dvips}
9 </dvips>
10 <*dvisvgm>
11   {l3backend-testphase-dvisvgm.def}{2022-11-03}{}
12   {LaTeX-PDF-management-testphase-bundle-backend-support: dvisvgm}
13 </dvisvgm>
14 <*luatex>
15   {l3backend-testphase-luatex.def}{2022-11-03}{}
16   {LaTeX-PDF-management-testphase-bundle-backend-support: PDF output (LuaTeX)}
17 </luatex>
18 <*pdftex>
19   {l3backend-testphase-pdftex.def}{2022-11-03}{}
20   {LaTeX-PDF-management-testphase-bundle-backend-support: PDF output (pdfTeX)}
21 </pdftex>
22 <*xdvipdfmx>
23   {l3backend-testphase-xetex.def}{2022-11-03}{}
24   {LaTeX-PDF-management-testphase-bundle-backend-support: XeTeX}
25 </xdvipdfmx>
```

1.1 Crossreferences

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

```
26 <@@=pdf>
27 <*drivers>
```

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

```

28 \RequirePackage{l3ref-tmp}
29 \cs_generate_variant:Nn \ref_label:nn {en}
30 \cs_generate_variant:Nn \ref_value:nn {en}
31 \cs_new_protected:Npn \__pdf_backend_ref_label:nn #1 #2
32 {
33   \@bsphack
34   \ref_label:nn{#1}{abspage}
35   \@esphack
36 }
37 \cs_new:Npn \__pdf_backend_ref_value:nn #1 #2
38 {
39   \ref_value:nn{#1}{#2}
40 }
41 \cs_generate_variant:Nn \__pdf_backend_ref_label:nn {en}
42 \cs_generate_variant:Nn \__pdf_backend_ref_value:nn {en}
43 </drivers>

```

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

```

44 <*dvipdfmx |xdvipdfmx>
45   \__kernel_backend_literal:x { dvipdfmx:config-C~ 0x0010 }
46 </dvipdfmx |xdvipdfmx>

```

```

\g__pdf_tmpa_prop      Some scratch variables
  \l__pdf_tmpa_tl
\l__pdf_backend_tmpa_box
47 <*drivers>
48 \prop_new:N \g__pdf_tmpa_prop
49 \tl_new:N \l__pdf_tmpa_tl
50 \box_new:N \l__pdf_backend_tmpa_box
51 \box_new:N \l__pdf_backend_tmpb_box
52 </drivers>

```

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

```

\g__pdf_backend_resourceid_int  a counter to create labels for the resources, a counter to number properties in bdc marks,
\g__pdf_backend_name_int        a counter for the \pdfpageref implementation.
\g__pdf_backend_page_int
53 <*drivers>
54 \int_new:N \g__pdf_backend_resourceid_int
55 \int_new:N \g__pdf_backend_name_int
56 \int_new:N \g__pdf_backend_page_int
57 </drivers>

```

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

1.2 luacode

Load the lua code.

```

58 <*luatex>
59   \directlua { require("l3backend-testphase.lua") }
60 </luatex>

```

1.3 Converting unicode strings to a pdfname

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

```
61 <*pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
62 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
63 {
64   / \str_convert_pdfname:e { \text_expand:n { #1 } }
65 }
66 </pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
67 <*dvips>
68 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
69 {
70   ~ ( \text_expand:n { #1 } ) ~ cvn
71 }
72 </dvips>
```

1.4 Hooks

1.4.1 Add the “end run” hooks

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

```
73 <*pdftex | luatex>
74 % put in \@kernel@after@enddocument@afterlastpage
75 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
76 {
77   \g__kernel_pdfmanagement_end_run_code_tl
78 }
79 </pdftex | luatex>
80 <*dvipdfmx | xdvipdfmx>
81 % put in \@kernel@after@shipout@lastpage
82 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
83 {
84   \g__kernel_pdfmanagement_end_run_code_tl
85 }
86 </dvipdfmx | xdvipdfmx>
87 <*dvips>
88 % put in \@kernel@after@shipout@lastpage
89 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
90 {
91   \g__kernel_pdfmanagement_end_run_code_tl
92 }
93 </dvips>
```

1.4.2 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 ...

```
94 <*drivers>
95 \tl_if_exist:NTF \@kernel@after@shipout@background
96 {
97   \g@addto@macro \@kernel@before@shipout@background{\relax}
98   \g@addto@macro \@kernel@after@shipout@background
```

```

99     {
100     \g__kernel_pdfmanagement_thispage_shipout_code_tl
101     }
102   }
103   {
104     \hook_gput_code:nnn{shipout/background}{pdf}
105     {
106       \g__kernel_pdfmanagement_thispage_shipout_code_tl
107     }
108   }
109
110 </drivers>

```

1.5 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.

```

111 <*pdftex>
112 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
113   {
114     \tex_global:D \tex_pdfpagesattr:D { #1 }
115   }
116 </pdftex>
117 <*luatex>
118 %luatex: does it in lua
119 \sys_if_engine_luatex:T
120   {
121     \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
122     {
123       \tex_directlua:D
124       {
125         pdf.setpagesattributes( \_pdf_backend_luastring:n { #1 } )
126       }
127     }
128   }
129 </luatex>
130 <*dvips>
131 \cs_new_protected:Npx \_pdf_backend_Pages_primitive:n #1
132   {
133     \tex_special:D{ps:~[#1~/PAGES~pdfmark] %}
134   }
135 </dvips>
136 <*dviPDFmx | xdviPDFmx>
137 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
138   {
139     \_pdf_backend:n{put~@pages~<<#1>>}
140   }
141 </dviPDFmx | xdviPDFmx>
142 <*dvisvgm>
143 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
144   {}

```

145 \langle /dvisvgm \rangle

(End definition for `_pdf_backend_Pages_primitive:n`.)

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

```
\_pdf_backend_Page_primitive:n \_pdf_backend_Page_primitive:n is the primitive command to add something to the
\_pdf_backend_Page_gput:nn /Page dictionary. It works differently for the backends: pdftex and luatex overwrite
\_pdf_backend_Page_gremove:n existing content, dvips and dvipldtx are additive. luatex sets it in lua. The higher
\_pdf_backend_ThisPage_gput:nn level code has to take this into account. \_pdf_backend_Page_gput:nn stores default
\_pdf_backend_ThisPage_gpush:n values. \_pdf_backend_Page_gremove:n allows to remove a value. \_pdf_backend_
ThisPage_gput:nn adds a value to the current page. \_pdf_backend_ThisPage_
gpush:n merges the default and the current page values and add them to the dictionary
of the current page in  $\backslash$ g\_pdf_backend_thispage_shipout_tl.

146 % backend commands
147  $\langle$ *pdftex $\rangle$ 
148 %the primitive
149 \cs_new_protected:Npn \_pdf_backend_Page_primitive:n #1
150 {
151     \tex_global:D \tex_pdfpageattr:D { #1 }
152 }
153 % the command to store default values.
154 % Uses a prop with pdflatex + dvi,
155 % sets a lua table with luatex
156 \cs_new_protected:Npn \_pdf_backend_Page_gput:nn #1 #2 %key,value
157 {
158     \pdfdict_gput:nnn {g\_pdf_Core/Page}{ #1 }{ #2 }
159 }
160 % the command to remove a default value.
161 % Uses a prop with pdflatex + dvi,
162 % changes a lua table with luatex
163 \cs_new_protected:Npn \_pdf_backend_Page_gremove:n #1
164 {
165     \pdfdict_gremove:nn {g\_pdf_Core/Page}{ #1 }
166 }
167 % the command used in the document.
168 % direct call of the primitive special with dvips/dvipdtx
169 % \latelua: fill a page related table with luatex, merge it with the page
170 % table and push it directly
171 % write to aux and store in prop with pdflatex
172 \cs_new_protected:Npn \_pdf_backend_ThisPage_gput:nn #1 #2
173 {
174     %we need to know the page the resource should be added too.
175     \int_gincr:N\g\_pdf_backend_resourceid_int
176     \_pdf_backend_ref_label:en { l3pdf\int_use:N\g\_pdf_backend_resourceid_int }{abspage}
177     \tl_set:Nx \l\_pdf_tmpa_tl
178     {
179         \_pdf_backend_ref_value:en {l3pdf\int_use:N\g\_pdf_backend_resourceid_int}{abspage}
180     }
181     \pdfdict_if_exist:nF { g\_pdf_Core/backend_Page\l\_pdf_tmpa_tl}
182     {
183         \pdfdict_new:n { g\_pdf_Core/backend_Page\l\_pdf_tmpa_tl}
184     }

```

```

185     %backend_Page has no handler.
186     \pdfdict_gput:nnn {g__pdf_Core/backend_Page\l__pdf_tmpa_tl}{ #1 }{ #2 }
187 }
188 %the code to push the values, used in shipout
189 %merges the two props and then fills the register in pdflatex
190 %merges the two tables and then fills (in lua) in luatex
191 %issues the values stored in the global prop with dvi
192 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
193 {
194     \prop_gset_eq:Nc \g__pdf_tmpa_prop { \__kernel_pdfdict_name:n { g__pdf_Core/Page } }
195     \prop_if_exist:cT { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
196     {
197         \prop_map_inline:cn { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
198         {
199             \prop_gput:Nnn \g__pdf_tmpa_prop { ##1 }{ ##2 }
200         }
201     }
202     \exp_args:Nx \__pdf_backend_Page_primitive:n
203     {
204         \prop_map_function:NN \g__pdf_tmpa_prop \pdfdict_item:ne
205     }
206 }
207 </pdfTeX>
208 <*luatex>
209 % do we need to use some escaping for the values?????
210 \cs_new:Npn \__pdf_backend_luastring:n #1
211 {
212     "\tex_luaescapestring:D { \tex_unexpanded:D { #1 } }"
213 }
214 %not used, only there for consistency
215 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
216 {
217     \tex_latelua:D
218     {
219         pdf.setpageattributes(\__pdf_backend_luastring:n { #1 })
220     }
221 }
222 % the command to store default values.
223 % Uses a prop with pdflatex + dvi,
224 % sets a lua table with luatex
225 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
226 {
227     \tex_directlua:D
228     {
229         ltx.__pdf.backend_Page_gput
230         (
231             \__pdf_backend_luastring:n { #1 },
232             \__pdf_backend_luastring:n { #2 }
233         )
234     }
235 }
236 % the command to remove a default value.
237 % Uses a prop with pdflatex + dvi,
238 % changes a lua table with luatex

```

```

239 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
240 {
241   \tex_directlua:D
242   {
243     ltx.__pdf.backend_Page_gremove (\__pdf_backend_luastring:n { #1 })
244   }
245 }
246 % the command used in the document.
247 % direct call of the primitive special with dvips/dvipdfmx
248 % \lualua: fill a page related table with luatex, merge it with the page
249 % table and push it directly
250 % write to aux and store in prop with pdflatex
251 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
252 {
253   \tex_latelua:D
254   {
255     ltx.__pdf.backend_ThisPage_gput
256     (
257       tex.count["g_shipout_readonly_int"],
258       \__pdf_backend_luastring:n { #1 },
259       \__pdf_backend_luastring:n { #2 }
260     )
261     ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
262   }
263 }
264 %the code to push the values, used in shipout
265 %merges the two props and then fills the register in pdflatex
266 %merges the two tables (the one is probably still empty) and then fills (in lua) in luatex
267 %issues the values stored in the global prop with dvi
268 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
269 {
270   \tex_latelua:D
271   {
272     ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
273   }
274 }
275
276 </luatex>
277 < *dvipdfmx | xdvipdfmx >
278 %the primitive
279 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
280 {
281   \tex_special:D{pdf:-put~@thispage~<<#1>>}
282 }
283 % the command to store default values.
284 % Uses a prop with pdflatex + dvi,
285 % sets a lua table with luatex
286 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
287 {
288   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
289 }
290 % the command to remove a default value.
291 % Uses a prop with pdflatex + dvi,
292 % changes a lua table with luatex

```

```

293 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
294   {
295     \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
296   }
297   % the command used in the document.
298   % direct call of the primitive special with dvips/dvipdfmx
299   % \latalua: fill a page related table with luatex, merge it with the page
300   % table and push it directly
301   % write to aux and store in prop with pdflatex
302 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
303   {
304     \__pdf_backend_Page_primitive:n { /#1~#2 }
305   }
306   %the code to push the values, used in shipout
307   %merges the two props and then fills the register in pdflatex
308   %merges the two tables (the one is probably still empty)
309   % and then fills (in lua) in luatex
310   %issues the values stored in the global prop with dvi
311 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
312   {
313     \exp_args:Nx \__pdf_backend_Page_primitive:n
314     { \pdfdict_use:n { g__pdf_Core/Page} }
315   }
316   </dvipdfmx|xdvipdfmx>
317   <*dvips>
318 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
319   {
320     \tex_special:D{ps:-[ThisPage]<<#1>>~/PUT~pdfmark} %]
321   }
322   % the command to store default values.
323   % Uses a prop with pdflatex + dvi,
324   % sets a lua table with luatex
325 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
326   {
327     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
328   }
329   % the command to remove a default value.
330   % Uses a prop with pdflatex + dvi,
331   % changes a lua table with luatex
332 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
333   {
334     \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
335   }
336   % the command used in the document.
337   % direct call of the primitive special with dvips/dvipdfmx
338   % \latalua: fill a page related table with luatex, merge it with the page
339   % table and push it directly
340   % write to aux and store in prop with pdflatex
341 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
342   {
343     \__pdf_backend_Page_primitive:n { /#1~#2 }
344   }
345   %the code to push the values, used in shipout
346   %merges the two props and then fills the register in pdflatex

```



```

347 %merges the two tables (the one is probably still empty)
348 %and then fills (in lua) in luatex
349 %issues the values stored in the global prop with dvi
350 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
351 {
352   \exp_args:Nx \__pdf_backend_Page_primitive:n
353     { \pdfdict_use:n { g__pdf_Core/Page} }
354 }
355 </dvips>
356 <*dvisvgm>
357 % mostly only dummies ...
358 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
359 {}
360 % Uses a prop with pdflatex + dvi,
361 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
362 {
363   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
364 }
365 % the command to remove a default value.
366 % Uses a prop with pdflatex + dvi,
367 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
368 {
369   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
370 }
371 % the command used in the document.
372 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
373 {}
374 %the code to push the values, used in shipout
375 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
376 {}
377 </dvisvgm>

```

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

1.7 “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.

```

378 <*drivers>
379 \clist_const:Nn \c__pdf_backend_PageResources_clist
380 {
381   ExtGState,
382   ColorSpace,
383   Pattern,
384   Shading,
385 }
386 </drivers>

```

(End definition for `\c__pdf_backend_PageResources_clist`.)

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

`_pdf_backend_PageResources_gput:nnn` stores values for the page resources.
#1 : name of the resource (ExtGState, ColorSpace, Shading, Pattern)
#2 : a pdf name without slash
#3 : value

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

`_pdf_backend_PageResources_obj_gpush:`

```

387 <*pdfTeX | luatex>
388 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
389   {
390     \pdf_object_new:n {__pdf/Page/Resources/#1}
391     \cs_if_exist:NT \tex_directlua:D
392       {
393         \tex_directlua:D
394           {
395             ltx.__pdf.object["__pdf/Page/Resources/#1"]
396             =
397             "\_pdf_backend_object_ref:n{__pdf/Page/Resources/#1}"
398           }
399       }
400   }
401 </pdfTeX | luatex>

```

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

```

402 <*luatex>
403 \cs_new_protected:Npn \_pdf_backend_PageResources_gput:nnn #1 #2 #3
404   {
405     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
406     \tex_latelua:D{ltx.__pdf.Page.Resources.#1=true}
407     \tex_latelua:D
408       {
409         ltx.pdf.Page.Resources_gpush(tex.count["g_shipout_readonly_int"])
410       }
411   }
412 </luatex>
413 <*pdfTeX>
414 \cs_new_protected:Npn \_pdf_backend_PageResources_gput:nnn #1 #2 #3
415   {
416     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
417   }
418 </pdfTeX>

```

code for end of document code

```

419 <*pdfTeX | luatex>
420 \cs_new_protected:Npn \_pdf_backend_PageResources_obj_gpush:
421   {
422     \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
423       {
424         \prop_if_empty:cF
425           { \_kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1 } }
426           {
427             \pdf_object_write:nxx
428               { __pdf/Page/Resources/##1 } { dict }

```

```

429         { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1 }
430     }
431 }
432 }
433 </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.

```

434 <*dvipdfmx | xdvipdfmx>
435 <xdvipdfmx> \hook_gset_rule:nnn{shipout/firstpage}{l3backend-xetex}{after}{pdf}
436 <dvipdfmx> \hook_gset_rule:nnn{shipout/firstpage}{l3backend-dvipdfmx}{after}{pdf}
437 %
438 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
439 {
440   \pdf_object_new:n { __pdf/Page/Resources/#1 }
441   \hook_gput_code:nnn
442     {shipout/firstpage}
443     {pdf}
444     {\pdf_object_write:nnn { __pdf/Page/Resources/#1 } { dict } {}}
445 }
446 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1
447 {
448   \__pdf_backend:n {put~@resources~<<#1>>}
449 }
450 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
451 {
452   % this is not used for output, but there is a test if the resource is empty
453   \exp_args:Nnx
454   \prop_gput:cnn { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1 } }
455   { \str_convert_pdfname:n {#2} }{ #3 }
456   %objects are not filled with \pdf_object_write as this is not additive!
457   \__pdf_backend:x
458   {
459     put~\__pdf_backend_object_ref:n {__pdf/Page/Resources/#1}<<#2~#3>>
460   }
461 }
462
463 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
464 </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.

```

465 <*dvips>
466 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
467 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
468 { %only for the show command TEST!!
469   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
470 }
471 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
472 </dvips>

```

dvipsvgm unneeded, or no-op

```

473 <*dvisvgm>
474 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
475 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
476   { %only for the show command TEST!!
477     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
478   }
479 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
480 </dvisvgm>

```

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

1.7.1 Page resources /Properties + BDC operators

```

\__pdf_backend_bdc:nn \__pdf_backend_bdcobject:nn, \__pdf_backend_bdcobject:n,
\__pdf_backend_bdcobject:nn \__pdf_backend_bmc:n and \__pdf_backend_emc: are the backend command that cre-
\__pdf_backend_bdcobject:n ate the bdc/emc marker and store the properties. \__pdf_backend_PageResources_-
\__pdf_backend_bmc:n gpush:n outputs the /Properties and/or the other resources for the current page.
\__pdf_backend_emc:
\__pdf_backend_PageResources_gpush:n
481 % pdftex and luatex (and perhaps dvips ...) need to know if there are in a
482 % xform stream ...
483 <*drivers>
484 \bool_new:N \l__pdf_backend_xform_bool
485 </drivers>
486 <*dvips>
487 % dvips is easy: create an object, and reference it in the bdc
488 % ghostscript will then automatically replace it by a name
489 % and add the name to the /Properties dict
490 % special variant von accsupp
491 % https://chat.stackexchange.com/transcript/message/50831812#50831812
492 %
493 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
494   {
495     \__pdf_backend_pdfmark:x{/#1-<<#2>>~/BDC}
496   }
497 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
498   {
499     \__pdf_backend_pdfmark:x{/#1-\__pdf_backend_object_ref:n{#2}~/BDC}
500   }
501 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
502   {
503     \__pdf_backend_pdfmark:x{/#1-\__pdf_backend_object_last:~/BDC}
504   }
505 \cs_set_protected:Npn \__pdf_backend_emc:
506   {
507     \__pdf_backend_pdfmark:n{/EMC} %
508   }
509 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
510   {
511     \__pdf_backend_pdfmark:n{/#1~/BMC} %
512   }
513 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
514
515 </dvips>
516 <*dvisvgm>

```

```

517 % dvisvgm should do nothing
518 %
519 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
520 {}
521 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
522 {}
523 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
524 {}
525 \cs_set_protected:Npn \__pdf_backend_emc:
526 {}
527 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
528 {}
529 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
530
531 </dvisvgm>
532
533 % xetex has to create the entries in the /Properties manually
534 % (like the other backends)
535 % use pdfbase special
536 % https://chat.stackexchange.com/transcript/message/50832016#50832016
537 % the property is added to xform resources automatically,
538 % no need to worry about it.
539 <*dviPDFmx|xDviPDFmx>
540 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
541 {
542   \int_gincr:N \g__pdf_backend_name_int
543   \__kernel_backend_literal:x
544   {
545     pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
546   }
547   \__kernel_backend_literal:x
548   {
549     pdf:put~@resources~
550     <<
551       /Properties~
552       <<
553         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
554         \__pdf_backend_object_ref:n { #2 }
555       >>
556     >>
557   }
558 }
559 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span
560 {
561   \int_gincr:N \g__pdf_backend_name_int
562   \__kernel_backend_literal:x
563   {
564     pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
565   }
566   \__kernel_backend_literal:x
567   {
568     pdf:put~@resources~
569     <<
570     /Properties~

```

```

571         <<
572             /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
573             \__pdf_backend_object_last:
574         >>
575     >>
576     }
577 }
578 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
579 {
580     \__kernel_backend_literal:n {pdf:code~/#1~BMC} %pdfbase
581 }
582
583 %this require management
584 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
585 {
586     \pdf_object_unnamed_write:nn { dict }{ #2 }
587     \__pdf_backend_bdcobject:n { #1 }
588 }
589
590 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
591 {
592     \__kernel_backend_literal:n {pdf:code~ /#1~<<#2>>~BDC }
593 }
594
595 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
596 {
597     \bool_if:NTF \g__pdfmanagement_active_bool
598     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
599     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
600     \__pdf_backend_bdc:nn {#1}{#2}
601 }
602 \cs_set_protected:Npn \__pdf_backend_emc:
603 {
604     \__kernel_backend_literal:n {pdf:code~EMC} %pdfbase
605 }
606 % properties are handled automatically, but the other resources should be added
607 % at shipout
608 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
609 {
610     \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
611     {
612         \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
613         {
614             \__kernel_backend_literal:x
615             {
616                 pdf:put~@resources~
617                 <</##1~\__pdf_backend_object_ref:n {__pdf/Page/Resources/##1}>>
618             }
619         }
620     }
621 }
622 </dviptdpmx | xdvipdpmx>
623 % luatex + pdftex
624 <*luatex>

```

```

625 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
626 {
627   \int_gincr:N \g__pdf_backend_name_int
628   \exp_args:Nx\__kernel_backend_literal_page:n
629   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
630   \bool_if:NTF \l__pdf_backend_xform_bool
631   {
632     \exp_args:Nnx\pdfdict_gput:nnn
633     { g__pdf_Core/Xform/Resources/Properties }
634     { l3pdf\int_use:N\g__pdf_backend_name_int }
635     { \__pdf_backend_object_ref:n { #2 } }
636   }
637   {
638     \exp_args:Nx \tex_latelua:D
639     {
640       ltx.pdf.Page_Resources_Properties_gput
641       (
642         tex.count["g_shipout_readonly_int"],
643         "l3pdf\int_use:N\g__pdf_backend_name_int",
644         "\__pdf_backend_object_ref:n { #2 }"
645       )
646     }
647   }
648 }
649 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
650 {
651   \int_gincr:N \g__pdf_backend_name_int
652   \exp_args:Nx\__kernel_backend_literal_page:n
653   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
654   \bool_if:NTF \l__pdf_backend_xform_bool
655   {
656     \exp_args:Nnx\pdfdict_gput:nnn %no handler needed
657     { g__pdf_Core/Xform/Resources/Properties }
658     { l3pdf\int_use:N\g__pdf_backend_name_int }
659     { \__pdf_backend_object_last: }
660   }
661   {
662     \exp_args:Nx \tex_latelua:D
663     {
664       ltx.pdf.Page_Resources_Properties_gput
665       (
666         tex.count["g_shipout_readonly_int"],
667         "l3pdf\int_use:N\g__pdf_backend_name_int",
668         "\__pdf_backend_object_last:"
669       )
670     }
671   }
672 }
673 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
674 {
675   \__kernel_backend_literal_page:n { /#1~BMC }
676 }
677 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
678 {

```

```

679   \pdf_object_unnamed_write:nn { dict } { #2 }
680   \__pdf_backend_bdcobject:n { #1 }
681 }
682 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
683 {
684   \__kernel_backend_literal_page:n { /#1-<<#2>>~BDC }
685 }
686 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
687 {
688   \bool_if:NTF \g__pdfmanagement_active_bool
689     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
690     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
691     \__pdf_backend_bdc:nn {#1}{#2}
692 }
693 \cs_set_protected:Npn \__pdf_backend_emc:
694 {
695   \__kernel_backend_literal_page:n { EMC }
696 }
697
698 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
699 </luatex>
700 <*pdfTeX>
701 % pdfLaTeX is the most complicated as it has to go through the aux ...
702 % the push command is extended to take other resources too
703 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
704 {
705   \int_gincr:N \g__pdf_backend_name_int
706   \exp_args:Nx\__kernel_backend_literal_page:n
707     { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
708   % code to set the property ....
709   \int_gincr:N\g__pdf_backend_resourceid_int
710   \bool_if:NTF \l__pdf_backend_xform_bool
711     {
712       \exp_args:Nxxx\pdfdict_gput:nnn %no handler needed
713         { g__pdf_Core/Xform/Resources/Properties }
714         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
715         { \__pdf_backend_object_ref:n { #2 } }
716     }
717     {
718       \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
719       \tl_set:Nx \l__pdf_tmpa_tl
720         {
721           \__pdf_backend_ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
722         }
723       \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
724         {
725           \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
726         }
727       \exp_args:Nxxx\pdfdict_gput:nnn
728         { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
729         { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
730         { \__pdf_backend_object_ref:n{#2} }
731     }
732 }

```



```

733 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
734 {
735   \int_gincr:N \g__pdf_backend_name_int
736   \exp_args:Nx\__kernel_backend_literal_page:n
737   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
738   % code to set the property ....
739   \int_gincr:N\g__pdf_backend_resourceid_int
740   \bool_if:NTF \l__pdf_backend_xform_bool
741   {
742     \exp_args:Nnxx\pdfdict_gput:nnn
743     { g__pdf_Core/Xform/Resources/Properties }
744     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
745     { \__pdf_backend_object_last: }
746   }
747   {
748     \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
749     \tl_set:Nx \l__pdf_tmpa_tl
750     {
751       \__pdf_backend_ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
752     }
753     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
754     {
755       \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
756     }
757     \exp_args:Nnxx\pdfdict_gput:nnn
758     { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
759     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
760     { \__pdf_backend_object_last: }
761     %\pdfdict_show:n { g_backend_Page\l__pdf_tmpa_tl/Resources/Properties }
762   }
763 }
764 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
765 {
766   \__kernel_backend_literal_page:n { /#1-BMC }
767 }
768 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
769 {
770   \pdf_object_unnamed_write:nn { dict } { #2 }
771   \__pdf_backend_bdcobject:n { #1 }
772 }
773 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
774 {
775   \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
776 }
777 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
778 {
779   \bool_if:NTF \g__pdfmanagement_active_bool
780   {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
781   {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
782   \__pdf_backend_bdc:nn {#1}{#2}
783 }
784 \cs_set_protected:Npn \__pdf_backend_emc:
785 {
786   \__kernel_backend_literal_page:n { EMC }

```

```

787 }
788
789 \cs_new:Npn \__pdf_backend_PageResources_gpush_aux:n #1 %#1 ExtGState etc
790 {
791   \prop_if_empty:cF
792     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
793     {
794       \pdfdict_item:ne { #1 }{ \pdf_object_ref:n {__pdf/Page/Resources/#1}}
795     }
796 }
797
798 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
799 {
800   \exp_args:NNx \tex_global:D \tex_pdfpageresources:D
801   {
802     \prop_if_exist:cT
803       { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
804       {
805         /Properties~
806         <<
807         \prop_map_function:cN
808           { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Property
809             \pdfdict_item:ne
810             >>
811         }
812         %% add ExtGState etc
813         \clist_map_function:NN
814           \c__pdf_backend_PageResources_clist
815           \__pdf_backend_PageResources_gpush_aux:n
816       }
817   }
818
819 </pdftex>

```

(End definition for __pdf_backend_bdc:nn and others.)

1.8 “Catalog” & subdirectories (pdfcatalog)

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

1.8.1 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.

```

820 % pdflatex
821 <*pdftex>
822 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
823 {
824   \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
825   \tex_pdfnames:D {/#1~\pdf_object_ref_last:}
826 }

```

```

827 </pdfTeX>
828 <*luatex>
829 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
830 {
831     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
832     \tex_pdfextension:D~names~ {/#1~\pdf_object_ref_last:}
833 }
834 </luatex>
835 <*dviPDFmx | xdvipdfmx>
836 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
837 {
838     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
839     \__pdf_backend:x {put~@names~<</#1~\pdf_object_ref_last: >>}
840 }
841 </dviPDFmx | xdvipdfmx>
842
843 %dvips: noop
844 <*dvips>
845 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
846 </dvips>
847 %dvisvgm: noop
848 <*dvisvgm>
849 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
850 </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.

```

851 <*pdfTeX | luatex | dviPDFmx | xdvipdfmx>
852 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2 {}
853 </pdfTeX | luatex | dviPDFmx | xdvipdfmx>
854 <*dvips>
855 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
856 {
857     \__pdf_backend_pdfmark:x
858     {
859         /Name~#1~
860         /FS~#2~
861         /EMBED
862     }
863 }
864 </dvips>
865 <*dvisvgm>
866 %no op. Or is there any sensible use for it?
867 \cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
868 {}
869
870 </dvisvgm>

```

(End definition for `__pdf_backend_NamesEmbeddedFiles_add:nn`.)

1.8.2 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.

```
871 <*drivers>
872 \cs_new_protected:Npn \__pdf_backend_link_off: {}
873 \cs_new_protected:Npn \__pdf_backend_link_on: {}
874 </drivers>
875 <*pdftex>
876 \cs_if_exist:NT \pdfrunninglinkoff
877 {
878   \cs_set_protected:Npn \__pdf_backend_link_off:
879     {
880       \pdfrunninglinkoff
881     }
882   \cs_set_protected:Npn \__pdf_backend_link_on:
883     {
884       \pdfrunninglinkon
885     }
886 }
887 </pdftex>
888 <*luatex>
889 \int_compare:nNnT {\tex_luatexversion:D } > {112}
890 {
891   \cs_set_protected:Npn \__pdf_backend_link_off:
892     {
893       \pdfextension linkstate 1
894     }
895   \cs_set_protected:Npn \__pdf_backend_link_on:
896     {
897       \pdfextension linkstate 0
898     }
899 }
900 </luatex>
901 <*dviPDFmx | xdvipdfmx>
902 \cs_set_protected:Npn \__pdf_backend_link_off:
903   {
904     \__pdf_backend:n { noline }
905   }
906 \cs_set_protected:Npn \__pdf_backend_link_on:
907   {
908     \__pdf_backend:n { link }
909   }
910 </dviPDFmx | xdvipdfmx>
```

1.8.3 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
\__pdf_backend_xform_ref:n
```

```

911 <*pdfTeX>
912 \cs_new_protected:Npn \l__pdf_backend_xform_new:nnnn #1 #2 #3 #4
913 % #1 name
914 % #2 attributes
915 % #3 resources
916 % #4 content, not necessarily a box!
917 {
918   \hbox_set:Nn \l__pdf_backend_tmpa_box
919   {
920     \bool_set_true:N \l__pdf_backend_xform_bool
921     \prop_gc_clear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
922     #4
923   }
924   %store the dimensions
925   \tl_const:cx
926   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
927   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
928   \tl_const:cx
929   { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
930   { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
931   \tl_const:cx
932   { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
933   { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
934   %% do we need to test if #2 and #3 are empty??
935   \tex_immediate:D \tex_pdfxform:D
936   ~ attr ~ { #2 }
937   %% which other resources should be default? Is an argument actually needed?
938   ~ resources ~
939   {
940     #3
941     \int_compare:nNnT
942     { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
943     >
944     { 0 }
945     {
946       /Properties~
947       <<
948       \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
949       >>
950     }
951
952     \prop_if_empty:cF
953     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
954     {
955       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
956     }
957     \prop_if_empty:cF
958     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
959     {
960       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
961     }
962     \prop_if_empty:cF
963     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
964     {

```

```

965         /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
966     }
967     \prop_if_empty:cF
968     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
969     {
970         /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
971     }
972 }
973 \l__pdf_backend_tmpa_box
974 \int_const:cn
975 { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
976 { \tex_pdflastxform:D }
977 }
978
979 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
980 {
981     \tex_pdfrefxform:D
982     \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
983     \scan_stop:
984 }
985
986 \cs_new:Npn \__pdf_backend_xform_ref:n #1
987 {
988     \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
989 }
990 </pdftex>
991 <*luatex>
992 %luatex
993 %nearly identical but not completely ...
994 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
995 % #1 name
996 % #2 attributes
997 % #3 resources
998 % #4 content, not necessarily a box!
999 {
1000     \hbox_set:Nn \l__pdf_backend_tmpa_box
1001     {
1002         \bool_set_true:N \l__pdf_backend_xform_bool
1003         \prop_gclear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }
1004             #4
1005     }
1006     \tl_const:cx
1007     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1008     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1009     \tl_const:cx
1010     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1011     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1012     \tl_const:cx
1013     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1014     { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1015     %% do we need to test if #2 and #3 are empty??
1016     \tex_immediate:D \tex_pdfxform:D
1017     ~ attr ~ { #2 }
1018     %% which resources should be default? Is an argument actually needed?

```

```

1019 ~ resources ~
1020 {
1021 #3
1022 \int_compare:nNnT
1023   {\prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties
1024   >
1025   { 0 }
1026   {
1027     /Properties~
1028     <<
1029     \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1030     >>
1031   }
1032 \prop_if_empty:cF
1033   { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1034   {
1035     /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1036   }
1037 \prop_if_empty:cF
1038   { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1039   {
1040     /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1041   }
1042 \prop_if_empty:cF
1043   { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1044   {
1045     /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1046   }
1047 \prop_if_empty:cF
1048   { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1049   {
1050     /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1051   }
1052 }
1053 \l__pdf_backend_tmpa_box
1054 \int_const:cn
1055   { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1056   { \tex_pdflastxform:D }
1057 }
1058
1059 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 %protected as with xelatex
1060 {
1061   \tex_pdfrefxform:D \int_use:c
1062   {
1063     c__pdf_backend_xform_ \tl_to_str:n {#1} _int
1064   }
1065   \scan_stop:
1066 }
1067
1068 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1069   { \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R }
1070
1071 </luatex>
1072 <*dviptfm | xdvipdftm>

```

```

1073 % xetex
1074 % it needs a bit testing if it really works to set the box to 0 before the special ...
1075 % does it disturb viewing the xobject?
1076 % what happens with the resources (bdc)? (should work as they are specials too)
1077 % xetex requires that the special is in horizontal mode. This means it affects
1078 % typesetting. But we can no delay the whole form code to shipout
1079 % as the object reference and the size is often wanted on the current page.
1080 % so we need to allocate a box - but probably they won't be thousands xform
1081 % in a document so it shouldn't matter.
1082 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1083 % #1 name
1084 % #2 attributes
1085 % #3 resources
1086 % #4 content, not necessarily a box!
1087 {
1088   \int_gincr:N \g__pdf_backend_object_int
1089   \int_const:cn
1090     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1091     { \g__pdf_backend_object_int }
1092   \box_new:c { g__pdf_backend_xform_#1_box }
1093   \hbox_gset:cn { g__pdf_backend_xform_#1_box }
1094     {
1095       \bool_set_true:N \l__pdf_backend_xform_bool
1096       #4
1097     }
1098   \tl_const:cx
1099     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1100     { \tex_the:D \box_wd:c { g__pdf_backend_xform_#1_box } }
1101   \tl_const:cx
1102     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1103     { \tex_the:D \box_ht:c { g__pdf_backend_xform_#1_box } }
1104   \tl_const:cx
1105     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1106     { \tex_the:D \box_dp:c { g__pdf_backend_xform_#1_box } }
1107   \box_set_dp:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1108   \box_set_ht:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1109   \box_set_wd:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1110   \hook_gput_next_code:nn {shipout/background}
1111   {
1112     \mode_leave_vertical: %needed, the xform disappears without it.
1113     \__pdf_backend:x
1114     {
1115       bxobj ~ \__pdf_backend_xform_ref:n { #1 }
1116       \c_space_tl width ~ \pdfxform_wd:n { #1 }
1117       \c_space_tl height ~ \pdfxform_ht:n { #1 }
1118       \c_space_tl depth ~ \pdfxform_dp:n { #1 }
1119     }
1120     \box_use_drop:c { g__pdf_backend_xform_#1_box }
1121     \__pdf_backend:x {put ~ @resources ~<<#3>> }
1122     \__pdf_backend:x
1123     {
1124       put~ @resources ~
1125       <<
1126       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }

```



```

1127         >>
1128     }
1129     \__pdf_backend:x
1130     {
1131         put~ @resources ~
1132         <<
1133             /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1134         >>
1135     }
1136     \__pdf_backend:x
1137     {
1138         put~ @resources ~
1139         <<
1140             /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1141         >>
1142     }
1143     \__pdf_backend:x
1144     {
1145         put~ @resources ~
1146         <<
1147             /ColorSpace~
1148             \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1149         >>
1150     }
1151     \exp_args:Nx
1152     \__pdf_backend:x {exobj ~<<#2>>}
1153 }
1154 }
1155
1156
1157
1158 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1159 {
1160     @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1161 }
1162
1163 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1164 {
1165     \hbox_set:Nn \l__pdf_backend_tmpa_box
1166     {
1167         \__pdf_backend:x
1168         {
1169             uxobj~ \__pdf_backend_xform_ref:n { #1 }
1170         }
1171     }
1172     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1173     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1174     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1175     \box_use_drop:N \l__pdf_backend_tmpa_box
1176 }
1177 </dvipdfmx | xdvipdfmx>
1178 <*dvisvgm>
1179 % unclear what it should do!!
1180 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}

```

```

1181 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
1182 \cs_new:Npn \__pdf_backend_xform_ref:n {}
1183 </divisvgn>

```

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

```

1184 <*dvips>
1185 \tl_new:N \l__pdf_backend_xform_tmpwd_tl
1186 \tl_new:N \l__pdf_backend_xform_tmpdp_tl
1187 \tl_new:N \l__pdf_backend_xform_tmph_tl

1188 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 % #1 name, #2 attribute, #4
1189 {
1190   \int_gincr:N \g__pdf_backend_object_int
1191   \int_const:cn
1192     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1193     { \g__pdf_backend_object_int }
1194
1195   \hbox_set:Nn \l__pdf_backend_tmpa_box
1196     {
1197       \bool_set_true:N \l__pdf_backend_xform_bool
1198       \prop_gc_clear:c {\__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
1199       #4
1200     }
1201   %store the dimensions
1202   \tl_const:cx
1203     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1204     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1205   \tl_const:cx
1206     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1207     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1208   \tl_const:cx
1209     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1210     { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1211   %store content dimensions in DPI units (Dots) (code from issue 25)
1212   \tl_set:Nx \l__pdf_backend_xform_tmpwd_tl
1213     {
1214       \dim_to_decimal_in_sp:n{ \box_wd:N \l__pdf_backend_tmpa_box }~
1215       65536~div~72.27~div~DVImag~mul~Resolution~mul~
1216     }
1217   \tl_set:Nx \l__pdf_backend_xform_tmph_tl
1218     {
1219       \dim_to_decimal_in_sp:n{ \box_ht:N \l__pdf_backend_tmpa_box }~
1220       65536~div~72.27~div~DVImag~mul~VResolution~mul~
1221     }
1222   \tl_set:Nx \l__pdf_backend_xform_tmpdp_tl
1223     {
1224       \dim_to_decimal_in_sp:n{ \box_dp:N \l__pdf_backend_tmpa_box }~
1225       65536~div~72.27~div~DVImag~mul~VResolution~mul~
1226     }
1227   % mirror the box
1228   \%box_scale:Nnn \l__pdf_backend_tmpa_box {1} {-1}
1229   \hbox_set:Nn \l__pdf_backend_tmpb_box

```

```

1230 {
1231   \__kernel_backend_postscript:x
1232   {
1233     gsave~currentpoint~
1234     initclip~ % restore default clipping path (page device/whole page)
1235     clippath~pathbbox~newpath~pop~pop~
1236     \tl_use:N\l__pdf_backend_xform_tmpdp_tl~add~translate~
1237     mark~
1238     /_objdef~{ pdf.obj \int_use:N\g__pdf_backend_object_int }\c_space_tl~
1239     /BBox[
1240       0~
1241       \tl_use:N\l__pdf_backend_xform_tmplt_tl~
1242       \tl_use:N\l__pdf_backend_xform_tmpwd_tl~
1243       \tl_use:N\l__pdf_backend_xform_tmpdp_tl~
1244       neg
1245     ]
1246     \str_if_eq:eeF{#1}{}
1247     {
1248       product~(Distiller)~search~{pop~pop~pop~#2}{pop}ifelse~
1249     }
1250     /BP~pdfmark-1~-1~scale~neg~exch~neg~exch~translate
1251   }
1252   \box_use_drop:N\l__pdf_backend_tmpa_box
1253   \__kernel_backend_postscript:n
1254   {
1255     mark ~ /EP~pdfmark ~ grestore
1256   }
1257   \str_if_eq:eeF{#1}{}
1258   {
1259     \__kernel_backend_postscript:x
1260     {
1261       product~(Ghostscript)~search~
1262       {
1263         pop~pop~pop~
1264         mark~
1265         { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1266         ~<<#2>>~/PUT~pdfmark
1267       }{pop}ifelse
1268     }
1269   }
1270 }
1271 \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1272 \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1273 \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1274 \hook_gput_code:nnn {begindocument/end}{pdfxform}
1275 {
1276   \mode_leave_vertical:
1277   \box_use:N\l__pdf_backend_tmpb_box
1278 }
1279 }
1280
1281
1282 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1283 {

```

```

1284 \hbox_set:Nn \l__pdf_backend_tmpa_box
1285 {
1286   \__kernel_backend_postscript:x
1287   {
1288     gsave~currentpoint~translate~1~-1~scale~
1289     mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int }}~
1290     /SP~pdfmark ~ grestore
1291   }
1292 }
1293 \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1294 \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1295 \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1296 \box_use_drop:N \l__pdf_backend_tmpa_box
1297 }
1298 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1299 {
1300   { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1301 }
1302
1303 </dvips>
1304 <*drivers>
1305 %% all
1306 \prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
1307 {
1308   \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1309   { \prg_return_true: }
1310   { \prg_return_false:}
1311 }
1312 \prg_new_eq_conditional:NNn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
1313 { TF , T , F , p }
1314 </drivers>

```

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

1.9 Structure Destinations

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

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

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

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

`\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
1315 <*drivers>
1316 \tl_new:N \l_pdf_current_structure_destination_tl
1317 </drivers>
```

(End 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:nmnm -> \__pdf_backend_structure_destination:nmnm
\__pdf_backend_link_begin_goto:nmw -> \__pdf_backend_link_begin_structure_goto:nmw
```

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

`\pdf_activate_structure_destination:`

```
1318 <*drivers>
1319 \cs_new_protected:Npn \pdf_activate_structure_destination:
1320 {
1321   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_structure_destination:nn
1322   \cs_gset_eq:NN \__pdf_backend_destination:nmnm \__pdf_backend_structure_destination:nmnm
1323   \cs_gset_eq:NN \__pdf_backend_link_begin_goto:nmw \__pdf_backend_link_begin_structure_goto:nmw
1324 }
1325 </drivers>
```

(End 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.

```
1326 <*drivers>
1327 \cs_set_eq:NN \__pdf_backend_structure_destination:nn \__pdf_backend_destination:nn
1328 \cs_set_eq:NN \__pdf_backend_structure_destination:nmnm \__pdf_backend_destination:nmnm
1329 \cs_set_eq:NN \__pdf_backend_link_begin_structure_goto:nmw \__pdf_backend_link_begin_goto:nmw
1330 </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:

```
1331 <*xdvipdfmx | dvipdfmx>
1332 \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1333 {
1334   \__pdf_backend:x
```

```

1335 {
1336   dest ~ ( \exp_not:n {#1} )
1337   [
1338     @thispage
1339     \str_case:nnF {#2}
1340     {
1341       { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1342       { fit } { /Fit }
1343       { fitb } { /FitB }
1344       { fitbh } { /FitBH }
1345       { fitbv } { /FitBV ~ @xpos }
1346       { fith } { /FitH ~ @ypos }
1347       { fitv } { /FitV ~ @xpos }
1348       { fitr } { /Fit }
1349     }
1350     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1351   ]
1352 }

```

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.

```

1353 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1354 {
1355   \__pdf_backend:x
1356   {
1357     obj ~ @pdf.SDest.\exp_not:n{#1}
1358     [
1359       \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1360       \str_case:nnF {#2}
1361       {
1362         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1363         { fit } { /Fit }
1364         { fitb } { /FitB }
1365         { fitbh } { /FitBH }
1366         { fitbv } { /FitBV ~ @xpos }
1367         { fith } { /FitH ~ @ypos }
1368         { fitv } { /FitV ~ @xpos }
1369         { fitr } { /Fit }
1370       }
1371       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1372     ]
1373   }
1374 }
1375 }

```

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

```

1376 \cs_new_protected:Npn \__pdf_backend_structure_destination_aux:nnnn #1#2#3#4
1377 {
1378   \vbox_to_zero:n
1379   {
1380     \__kernel_kern:n {#4}
1381     \hbox:n
1382     {

```

```

1383         \_pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1384         \_pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1385     }
1386     \tex_vss:D
1387 }
1388 \_kernel_kern:n {#1}
1389 \vbox_to_zero:n
1390 {
1391     \_kernel_kern:n { -#3 }
1392     \hbox:n
1393     {
1394         \_pdf_backend:n
1395         {
1396             dest ~ (#2)
1397             [
1398                 @thispage
1399                 /FitR ~
1400                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1401                 @xpos ~ @ypos
1402             ]
1403         }

```

Here we add the structure destination to the same box

```

1404         \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1405     {
1406         \_pdf_backend:x
1407         {
1408             obj ~ @pdf.SDest.\exp_not:n{#2}
1409             [
1410                 \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_
1411                 /FitR ~
1412                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1413                 @xpos ~ @ypos
1414             ]
1415         }
1416     }
1417 }
1418 \tex_vss:D
1419 }
1420 \_kernel_kern:n { -#1 }
1421 }

```

And now we redefine the destination command:

```

1422 \cs_set_protected:Npn \_pdf_backend_structure_destination:nnnn #1#2#3#4
1423 {
1424     \exp_args:Ne \_pdf_backend_structure_destination_aux:nnnn
1425     { \dim_eval:n {#2} } {#1} {#3} {#4}
1426 }

```

At last the goto link.

```

1427 \cs_set_protected:Npn \_pdf_backend_link_begin_structure_goto:nw #1#2
1428 {
1429     \_pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD~@pdf.SDest.
1430 }
1431 </xdvipdfmx | dvipdfmx>

```

(End definition for `_pdf_backend_structure_destination:nn`.)

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

```
1432 <*pdftex>
1433 \bool_lazy_and:nnT
1434 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1435 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
1436 {
1437   \cs_set_protected:Npn \_pdf_backend_structure_destination:nn #1#2
1438   {
1439     \tex_pdfdest:D
1440     name {#1}
1441     \str_case:nnF {#2}
1442     {
1443       { xyz } { xyz }
1444       { fit } { fit }
1445       { fitb } { fitb }
1446       { fitbh } { fitbh }
1447       { fitbv } { fitbv }
1448       { fith } { fith }
1449       { fitv } { fitv }
1450       { fitr } { fitr }
1451     }
1452     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1453     \scan_stop:
1454     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1455     {
1456       \tex_pdfdest:D
1457       struct~
1458       \int_use:c
1459       { c_pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1460       name {#1}
1461       \str_case:nnF {#2}
1462       {
1463         { xyz } { xyz }
1464         { fit } { fit }
1465         { fitb } { fitb }
1466         { fitbh } { fitbh }
1467         { fitbv } { fitbv }
1468         { fith } { fith }
1469         { fitv } { fitv }
1470         { fitr } { fitr }
1471       }
1472       { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1473       \scan_stop:
1474     }
1475   }
1476   \cs_set_protected:Npn \_pdf_backend_destination:nnnn #1#2#3#4
1477   {
1478     \tex_pdfdest:D
1479     name {#1}
1480     fitr ~
1481     width \dim_eval:n {#2} ~
1482     height \dim_eval:n {#3} ~
1483     depth \dim_eval:n {#4} \scan_stop:
```



```

1484     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1485     {
1486         \tex_pdfdest:D
1487         struct~
1488         \int_use:c
1489         { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_
1490         name {#1}
1491         fitr ~
1492         width \dim_eval:n {#2} ~
1493         height \dim_eval:n {#3} ~
1494         depth \dim_eval:n {#4} \scan_stop:
1495     }
1496 }
1497 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1498 {
1499     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1500 }
1501 }
1502 </pdfTeX>

```

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

```

1503 <*luatex>
1504 \int_compare:nNnT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1505 {
1506     \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1507     {
1508         \tex_pdfextension:D dest
1509         name {#1}
1510         \str_case:nnF {#2}
1511         {
1512             { xyz } { xyz }
1513             { fit } { fit }
1514             { fitb } { fitb }
1515             { fitbh } { fitbh }
1516             { fitbv } { fitbv }
1517             { fith } { fith }
1518             { fitv } { fitv }
1519             { fitr } { fitr }
1520         }
1521         { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1522         \scan_stop:
1523     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1524     {
1525         \tex_pdfextension:D dest
1526         struct~
1527         \int_use:c
1528         { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
1529         name {#1}
1530         \str_case:nnF {#2}
1531         {
1532             { xyz } { xyz }
1533             { fit } { fit }
1534             { fitb } { fitb }
1535             { fitbh } { fitbh }
1536             { fitbv } { fitbv }

```

```

1537         { fith } { fith }
1538         { fitv } { fitv }
1539         { fitr } { fitr }
1540     }
1541     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1542     \scan_stop:
1543 }
1544 }
1545 \cs_set_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
1546 {
1547     \tex_pdfextension:D dest
1548     name {#1}
1549     fitr ~
1550     width \dim_eval:n {#2} ~
1551     height \dim_eval:n {#3} ~
1552     depth \dim_eval:n {#4} \scan_stop:
1553     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1554     {
1555         \tex_pdfextension:D dest
1556         struct~
1557         \int_use:c
1558         { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n { \l_pdf_current_structure_
1559         name {#1}
1560         fitr ~
1561         width \dim_eval:n {#2} ~
1562         height \dim_eval:n {#3} ~
1563         depth \dim_eval:n {#4} \scan_stop:
1564     }
1565 }
1566 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1567 {
1568     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1569 }
1570 }
1571 </luatex>

```

1.10 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

```

1572 <*drivers>
1573 \cs_new_protected:Npn \__pdf_backend_set_regression_data:
1574 {
1575     \sys_gset_rand_seed:n{1000}
1576     \pdfmanagement_add:nnn{Info}{Creator}{(TeX)}
1577 </drivers>
1578 <*dvips>
1579     \__kernel_backend_literal:e{!~<</DocumentUUID~(DocumentUUID)>>~setpagedevice}
1580     \__kernel_backend_literal:e{!~<</InstanceUUID~(InstanceUUID)>>~setpagedevice}
1581 </dvips>
1582 <*dviPDFmx>
1583     \pdfmanagement_add:nnn{Info}{Producer}{(dviPDFmx)}

```

```

1584   \__kernel_backend_literal:e
1585   {pdf:trailerid [~
1586     <00112233445566778899aabbccddeeff>~
1587     <00112233445566778899aabbccddeeff>~
1588   ]}
1589 </dviPDFmx>
1590 <*xviPDFmx>
1591   \pdfmanagement_add:nnn{Info}{Producer}{(xetex)}
1592   \__kernel_backend_literal:e
1593   {pdf:trailerid [~
1594     <00112233445566778899aabbccddeeff>~
1595     <00112233445566778899aabbccddeeff>~
1596   ]}
1597 </xviPDFmx>
1598 <*pdftex>
1599   \pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX)}
1600   \tex_pdfsuppressptexinfo:D 7 \scan_stop:
1601   \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
1602 </pdftex>
1603 <*luatex>
1604   \pdfmanagement_add:nnn{Info}{Producer}{(LuaTeX)}
1605   \tex_pdfvariable:D suppressoptionalinfo 7\relax
1606   \tex_pdfvariable:D trailerid
1607   {[~
1608     <2350CAD05F8A7AF0AA4058486855344F>~
1609     <2350CAD05F8A7AF0AA4058486855344F>~
1610   ]}
1611 </luatex>
1612 <*drivers>
1613   \pdfmanagement_add:nnn{Info}{CreationDate}{(D:20010101205959-00'00')}
1614   \pdfmanagement_add:nnn{Info}{ModDate}{(D:20010101205959-00'00')}
1615   \AddToDocumentProperties[document]{pdfcreationdate}{D:20010101205959-00'00'}
1616   \AddToDocumentProperties[document]{pdfmoddate}{D:20010101205959-00'00'}
1617   \AddToDocumentProperties[hyperref]{pdfmetadate}{D:20010101205959-00'00'}
1618   \AddToDocumentProperties[hyperref]{pdfdate}{D:20010101205959-00'00'}
1619   \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-8c19-6237d832f
1620   }
1621 </drivers>

```

1.11 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`:

```

1622 <*luatex>
1623 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1624 {
1625   \tex_immediate:D \tex_pdfextension:D obj ~uncompressed~

```

```

1626     \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1627   }
1628 </luatex>
1629 <*pdfTeX>
1630 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1631   {
1632     \group_begin:
1633     \tex_pdfcompresslevel:D 0 \scan_stop:
1634     \tex_immediate:D \tex_pdfobj:D
1635     \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1636     \group_end:
1637   }
1638 </pdfTeX>
1639 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1640 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1641   {
1642     \pdf_object_unnamed_write:nn {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
1643   }
1644 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>

```

1.12 lua code for luatex

```

1645 <*lua>
1646 ltx= ltx or {}
1647 ltx.__pdf = ltx.__pdf or {}
1648 ltx.__pdf.Page = ltx.__pdf.Page or {}
1649 ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
1650 ltx.__pdf.Page.Resources = ltx.__pdf.Page.Resources or {}
1651 ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
1652 ltx.__pdf.Page.Resources.List={"ExtGState","ColorSpace","Pattern","Shading"}
1653 ltx.__pdf.object = ltx.__pdf.object or {}
1654
1655 ltx.pdf= ltx.pdf or {} -- for "public" functions
1656
1657 local __pdf = ltx.__pdf
1658 local pdf = pdf
1659
1660 local function __pdf_backend_Page_gput (name,value)
1661   __pdf.Page.dflt[name]=value
1662 end
1663
1664 local function __pdf_backend_Page_gremove (name)
1665   __pdf.Page.dflt[name]=nil
1666 end
1667
1668 local function __pdf_backend_Page_gclear ()
1669   __pdf.Page.dflt={}
1670 end
1671
1672 local function __pdf_backend_ThisPage_gput (page,name,value)
1673   __pdf.Page[page] = __pdf.Page[page] or {}
1674   __pdf.Page[page][name]=value
1675 end
1676

```

```

1677 local function __pdf_backend_ThisPage_gpush (page)
1678   local token=""
1679   local t = {}
1680   local tkeys= {}
1681   for name,value in pairs(__pdf.Page.dflt) do
1682     t[name]=value
1683   end
1684   if __pdf.Page[page] then
1685     for name,value in pairs(__pdf.Page[page]) do
1686       t[name] = value
1687     end
1688   end
1689   -- sort the table to get reliable test files.
1690   for name,value in pairs(t) do
1691     table.insert(tkeys,name)
1692   end
1693   table.sort(tkeys)
1694   for _,name in ipairs(tkeys) do
1695     token = token .. "/"..name.." "..t[name]
1696   end
1697   return token
1698 end
1699
1700 function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly"]
1701   __pdf_backend_ThisPage_gput (page,name,value)
1702 end
1703
1704 function ltx.__pdf.backend_ThisPage_gpush (page)
1705   pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
1706 end
1707
1708 function ltx.__pdf.backend_Page_gput (name,value)
1709   __pdf_backend_Page_gput (name,value)
1710 end
1711
1712 function ltx.__pdf.backend_Page_gremove (name)
1713   __pdf_backend_Page_gremove (name)
1714 end
1715
1716 function ltx.__pdf.backend_Page_gclear ()
1717   __pdf_backend_Page_gclear ()
1718 end
1719
1720
1721 local Properties = ltx.__pdf.Page.Resources.Properties
1722 local ResourceList= ltx.__pdf.Page.Resources.List
1723 local function __pdf_backend_PageResources_gpush (page)
1724   local token=""
1725   if Properties[page] then
1726     -- we sort the table, so that the pdf test works
1727     local t = {}
1728     for name,value in pairs (Properties[page]) do
1729       table.insert (t,name)
1730     end

```

```

1731 table.sort (t)
1732 for _,name in ipairs(t) do
1733   token = token .. "/"..name.." ".. Properties[page][name]
1734 end
1735 token = "/Properties <<"..token..">>"
1736 end
1737 for i,name in ipairs(ResourceList) do
1738   if ltx.__pdf.Page.Resources[name] then
1739     token = token .. "/"..name.." "..ltx.pdf.object_ref("__pdf/Page/Resources/"..name)
1740   end
1741 end
1742 return token
1743 end
1744
1745 -- the function is public, as I probably need it in tagpdf too ...
1746 function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
1747 Properties[page] = Properties[page] or {}
1748 Properties[page][name]=value
1749 pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
1750 end
1751
1752 function ltx.pdf.Page_Resources_gpush(page)
1753 pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
1754 end
1755
1756 function ltx.pdf.object_ref (objname)
1757 if ltx.__pdf.object[objname] then
1758   local ref= ltx.__pdf.object[objname]
1759   return ref
1760 else
1761   return "false"
1762 end
1763 end
1764 </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		box commands:
<code>\AddToDocumentProperties</code>		<code>\box_dp:N</code> .. 933, 1014, 1106, 1210, 1224
..... 1615, 1616, 1617, 1618, 1619		<code>\box_ht:N</code> .. 930, 1011, 1103, 1207, 1219
B		<code>\box_new:N</code> 50, 51, 1092
bool commands:		<code>\box_scale:Nnn</code> 1228
<code>\bool_if:NTF</code>		<code>\box_set_dp:Nn</code> 1107, 1174, 1271, 1295
.... 597, 630, 654, 688, 710, 740, 779		<code>\box_set_ht:Nn</code> 1108, 1173, 1272, 1294
<code>\bool_lazy_and:nnTF</code>	1433	<code>\box_set_wd:Nn</code> 1109, 1172, 1273, 1293
<code>\bool_new:N</code>	484	<code>\box_use:N</code>
<code>\bool_set_true:N</code> 920, 1002, 1095, 1197		1277
		<code>\box_use_drop:N</code> 1120, 1175, 1252, 1296
		<code>\box_wd:N</code> .. 927, 1008, 1100, 1204, 1214

C	
clist commands:	
\clist_const:Nn	379
\clist_map_function:NN	813
\clist_map_inline:Nn	388, 422, 438, 610
cs commands:	
\cs_generate_variant:Nn	29, 30, 41, 42
\cs_gset_eq:NN	598, 599, 689, 690, 780, 781, 1321, 1322, 1323
\cs_if_exist:NTF	391, 876
\cs_new:Npn	37, 62, 68, 210, 789, 986, 1068, 1158, 1182, 1298
\cs_new_protected:Npn	31, 112, 121, 137, 143, 149, 156, 163, 172, 192, 215, 225, 239, 251, 268, 279, 286, 293, 302, 311, 318, 325, 332, 341, 350, 358, 361, 367, 372, 375, 403, 414, 420, 446, 450, 463, 466, 467, 471, 474, 475, 479, 513, 529, 608, 698, 798, 822, 829, 836, 845, 849, 852, 855, 867, 872, 873, 912, 979, 994, 1059, 1082, 1163, 1180, 1181, 1188, 1282, 1319, 1376, 1573, 1623, 1630, 1640
\cs_new_protected:Npx	131
\cs_set_eq:NN	1327, 1328, 1329
\cs_set_protected:Npn	493, 497, 501, 505, 509, 519, 521, 523, 525, 527, 540, 559, 578, 584, 590, 595, 602, 625, 649, 673, 677, 682, 686, 693, 703, 733, 764, 768, 773, 777, 784, 878, 882, 891, 895, 902, 906, 1332, 1422, 1427, 1437, 1476, 1497, 1506, 1545, 1566
D	
dim commands:	
\dim_eval:n	1425, 1481, 1482, 1483, 1492, 1493, 1494, 1550, 1551, 1552, 1561, 1562, 1563
\dim_to_decimal_in_sp:n	1214, 1219, 1224
\c_zero_dim	1107, 1108, 1109, 1271, 1272, 1273
\directlua	59, 1504
E	
exp commands:	
\exp_args:Ne	1353, 1359, 1404, 1410, 1424, 1454, 1459, 1484, 1489, 1523, 1528, 1553, 1558
\exp_args:NNx	800
\exp_args:Nnx	453, 632, 656
\exp_args:Nnxx	712, 727, 742, 757
\exp_args:Nx	202, 313, 352, 628, 638, 652, 662, 706, 736, 1151
\exp_not:n	1336, 1357, 1408
F	
fp commands:	
\fp_eval:n	1350, 1371, 1452, 1472, 1521, 1541
G	
group commands:	
\group_begin:	1632
\group_end:	1636
H	
hbox commands:	
\hbox:n	1381, 1392
\hbox_gset:Nn	1093
\hbox_set:Nn	918, 1000, 1165, 1195, 1229, 1284
hook commands:	
\hook_gput_code:nnn	104, 441, 1274
\hook_gput_next_code:nn	1110
\hook_gset_rule:nnnn	435, 436
I	
int commands:	
\int_compare:nNnTF	889, 941, 1022, 1504
\int_compare_p:nNn	1434, 1435
\int_const:Nn	974, 1054, 1089, 1191
\int_gincr:N	175, 542, 561, 627, 651, 705, 709, 735, 739, 1088, 1190
\int_if_exist:NTF	1308
\int_new:N	54, 55, 56
\int_use:N	176, 179, 545, 553, 564, 572, 629, 634, 643, 653, 658, 667, 707, 714, 718, 721, 729, 737, 744, 748, 751, 759, 982, 988, 1061, 1069, 1160, 1238, 1265, 1289, 1300, 1458, 1488, 1527, 1557
K	
kernel internal commands:	
__kernel_backend_literal:n	45, 543, 547, 562, 566, 580, 592, 604, 614, 1579, 1580, 1584, 1592
__kernel_backend_literal_page:n	628, 652, 675, 684, 695, 706, 736, 766, 775, 786
__kernel_backend_postscript:n	1231, 1253, 1259, 1286
__kernel_kern:n	1380, 1388, 1391, 1420
__kernel_pdf_name_from_unicode_e:n	62, 68

<p> <code>__kernel_pdfdict_name:n</code> 194, 195, 197, 425, 454, 612, 792, 803, 808, 921, 942, 953, 958, 963, 968, 1003, 1023, 1033, 1038, 1043, 1048, 1198 <code>\g__kernel_pdfmanagement_end_-</code> <code> run_code_tl</code> 77, 84, 91 <code>\g__kernel_pdfmanagement_-</code> <code> thispage_shipout_code_tl</code> 100, 106 </p> <p style="text-align: center;">L</p> <p> latelua commands: <code>\latelua:</code> 169, 248, 299, 338 </p> <p style="text-align: center;">M</p> <p> mode commands: <code>\mode_leave_vertical:</code> . . . 1112, 1276 </p> <p style="text-align: center;">P</p> <p> pdf commands: <code>\pdf_activate_structure_destination:</code> 1318, 1319 <code>\l_pdf_current_structure_-</code> <code> destination_tl</code> 1315, 1353, 1359, 1404, 1410, 1454, 1459, 1484, 1489, 1523, 1528, 1553, 1558 <code>\pdf_object_if_exist:nTF</code> 1353, 1404, 1454, 1484, 1523, 1553 <code>\pdf_object_new:n</code> 390, 440 <code>\pdf_object_ref:n</code> . . . 794, 955, 960, 965, 970, 1035, 1040, 1045, 1050, 1126, 1133, 1140, 1148, 1359, 1410 <code>\pdf_object_ref_last:</code> . . . 825, 832, 839 <code>\pdf_object_unnamed_write:nn</code> 586, 679, 770, 824, 831, 838, 1642 <code>\pdf_object_write</code> 456 <code>\pdf_object_write:nnn</code> 427, 444 pdf internal commands: <code>__pdf_backend:n</code> 139, 448, 457, 839, 904, 908, 1113, 1121, 1122, 1129, 1136, 1143, 1152, 1167, 1334, 1355, 1383, 1384, 1394, 1406 <code>__pdf_backend_bdc:nn</code> 12, 481, 493, 519, 595, 598, 599, 600, 686, 689, 690, 691, 777, 780, 781, 782 <code>__pdf_backend_bdc_contobj:nn</code> 584, 598, 677, 689, 768, 780 <code>__pdf_backend_bdc_contstream:nn</code> 590, 599, 682, 690, 773, 781 <code>__pdf_backend_bdcobject:n</code> 12, 481, 501, 523, 559, 587, 649, 680, 733, 771 <code>__pdf_backend_bdcobject:nn</code> 12, 481, 497, 521, 540, 625, 703 <code>__pdf_backend_bmc:n</code> 12, 481, 509, 527, 578, 673, 764 </p>	<p> <code>__pdf_backend_catalog_gput:nn</code> . . . 18 <code>__pdf_backend_destination:nn</code> 1321, 1327 <code>__pdf_backend_destination:nnnn</code> 1322, 1328, 1476, 1545 <code>__pdf_backend_emc:</code> 12, 481, 505, 525, 602, 693, 784 <code>__pdf_backend_link_begin:n</code> . . . 1429 <code>__pdf_backend_link_begin:nnnw</code> 1499, 1568 <code>__pdf_backend_link_begin_-</code> <code> goto:nnw</code> 1323, 1329 <code>__pdf_backend_link_begin_-</code> <code> structure_goto:nnw</code> 1323, 1329, 1427, 1497, 1566 <code>__pdf_backend_link_off:</code> 872, 878, 891, 902 <code>__pdf_backend_link_on:</code> 873, 882, 895, 906 <code>__pdf_backend_luastring:n</code> 125, 210, 219, 231, 232, 243, 258, 259 <code>__pdf_backend_metadata_stream:n</code> 1623, 1630, 1640 <code>\g__pdf_backend_name_int</code> 53, 542, 545, 553, 561, 564, 572, 627, 629, 634, 643, 651, 653, 658, 667, 705, 707, 735, 737 <code>__pdf_backend_Names_gpush:nn</code> 822, 829, 836, 845, 849 <code>__pdf_backend_NamesEmbeddedFiles_-</code> <code> add:nn</code> 851, 852, 855, 867 <code>\g__pdf_backend_object_int</code> 1088, 1091, 1190, 1193, 1238 <code>__pdf_backend_object_last:</code> 503, 573, 659, 668, 745, 760 <code>__pdf_backend_object_ref:n</code> 397, 459, 499, 554, 617, 635, 644, 715, 730 <code>__pdf_backend_object_write:nn</code> 1626, 1635 <code>__pdf_backend_Page_gput:nn</code> 5, 146, 156, 225, 286, 325, 361 <code>__pdf_backend_Page_gremove:n</code> 5, 146, 163, 239, 293, 332, 367 <code>\g__pdf_backend_page_int</code> 53 <code>__pdf_backend_Page_primitive:n</code> 5, 146, 149, 202, 215, 279, 304, 313, 318, 343, 352, 358 <code>__pdf_backend_PageResources:n</code> 446, 466, 474 <code>\c__pdf_backend_PageResources_-</code> <code> clist</code> . . . 378, 388, 422, 438, 610, 814 <code>__pdf_backend_PageResources_-</code> <code> gpush:n</code> 12, 481, 513, 529, 608, 698, 798 </p>
--	---

__pdf_backend_PageResources_- gpush_aux:n	789, 815	\l__pdf_backend_xform_tmpwd_tl	1185, 1212, 1242
__pdf_backend_PageResources_- gput:nnn	387, 403, 414, 450, 467, 475	__pdf_backend_xform_use:n	911, 979, 1059, 1163, 1181, 1282
__pdf_backend_PageResources_- obj_gpush:	387, 420, 463, 471, 479	\g__pdf_tmpa_prop . . .	47, 194, 199, 204
__pdf_backend_Pages_primitive:n	111, 112, 121, 131, 137, 143	\l__pdf_tmpa_tl	47, 177, 181, 183, 186, 719, 723, 725, 728, 749, 753, 755, 758, 761
__pdf_backend_pdfmark:n	495, 499, 503, 507, 511, 857	pdfdict commands:	
__pdf_backend_ref_label:nn	31, 41, 176, 718, 748	\pdfdict_gput:nnn	158, 186, 288, 327, 363, 405, 416, 469, 477, 632, 656, 712, 727, 742, 757
__pdf_backend_ref_value:nn	37, 42, 179, 721, 751	\pdfdict_gremove:nn	165, 295, 334, 369
\g__pdf_backend_resourceid_int	53, 175, 176, 179, 709, 714, 718, 721, 729, 739, 744, 748, 751, 759	\pdfdict_if_exist:nTF . . .	181, 723, 753
__pdf_backend_set_regression_- data:	1573	\pdfdict_item:nn	204, 794, 809
__pdf_backend_structure_- destination:nn	1321, 1327, 1331, 1332, 1437, 1506	\pdfdict_new:n	183, 725, 755
__pdf_backend_structure_- destination:nnnn	1322, 1328, 1422	\pdfdict_show:n	761
__pdf_backend_structure_- destination_aux:nnnn	1376, 1424	\pdfdict_use:n	314, 353, 429, 948, 1029
__pdf_backend_ThisPage_gpush:n	5, 146, 192, 268, 311, 350, 375	\pdfextension	893, 897
__pdf_backend_ThisPage_gput:nn	5, 146, 172, 251, 302, 341, 372	pdfmanagement commands:	
\g__pdf_backend_thispage_- shipout_tl	5	\pdfmanagement_add:nnn	1576, 1583, 1591, 1599, 1604, 1613, 1614
\l__pdf_backend_tmpa_box	47, 918, 927, 930, 933, 973, 1000, 1008, 1011, 1014, 1053, 1165, 1172, 1173, 1174, 1175, 1195, 1204, 1207, 1210, 1214, 1219, 1224, 1228, 1252, 1284, 1293, 1294, 1295, 1296	pdfmanagement internal commands:	
\l__pdf_backend_tmpb_box	51, 1229, 1271, 1272, 1273, 1277	\g__pdfmanagement_active_bool	597, 688, 779
\l__pdf_backend_xform_bool	484, 630, 654, 710, 740, 920, 1002, 1095, 1197	\pdfnames	18
__pdf_backend_xform_if_exist:n	1306, 1312	\pdfpageref	2
__pdf_backend_xform_new:nnnn	911, 912, 994, 1082, 1180, 1188	\pdfrunninglinkoff	876, 880
__pdf_backend_xform_ref:n	911, 986, 1068, 1115, 1158, 1169, 1182, 1298	\pdfrunninglinkon	884
\l__pdf_backend_xform_tmpdp_tl	1186, 1222, 1236, 1243	\pdftrailerid	1601
\l__pdf_backend_xform_tmplt_tl	1187, 1217, 1241	pdfxform commands:	
		\pdfxform_dp:n	1118, 1174, 1295
		\pdfxform_ht:n	1117, 1173, 1294
		\pdfxform_if_exist:n	1312
		\pdfxform_wd:n	1116, 1172, 1293
		prg commands:	
		\prg_new_conditional:Npnn	1306
		\prg_new_eq_conditional:NNn	1312
		\prg_return_false:	1310
		\prg_return_true:	1309
		prop commands:	
		\prop_count:N	942, 1023
		\prop_gclear:N	921, 1003, 1198
		\prop_gput:Nnn	199, 454
		\prop_gset_eq:NN	194
		\prop_if_empty:NTF	424, 612, 791, 952, 957, 962, 967, 1032, 1037, 1042, 1047
		\prop_if_exist:NTF	195, 802
		\prop_map_function:NN	204, 807
		\prop_map_inline:Nn	197
		\prop_new:N	48
		\ProvidesExplFile	1

R	
ref commands:	
\ref_label:nn	29, 34
\ref_value:nn	30, 39
\relax	97, 1605
\RequirePackage	28
S	
scan commands:	
\scan_stop:	983, 1065, 1453, 1473, 1483, 1494, 1522, 1542, 1552, 1563, 1600, 1633
str commands:	
\str_case:nnTF	1339, 1360, 1441, 1461, 1510, 1530
\str_convert_pdfname:n	64, 455
\str_if_eq:nnTF	1246, 1257
sys commands:	
\sys_gset_rand_seed:n	1575
\sys_if_engine luatex:TF	119
T	
\TeX and $\LaTeX 2_{\epsilon}$ commands:	
\@bsphack	33
\@esphack	35
\@kernel@after@enddocument@afterlastpage	74, 75
\@kernel@after@shipout@background	95, 98
\@kernel@after@shipout@lastpage	81, 82, 88, 89
\@kernel@before@shipout@background	97
\g@addto@macro	97, 98
tex commands:	
\tex_directlua:D	123, 227, 241, 391, 393
\tex_global:D	114, 151, 800
\tex_immediate:D	935, 1016, 1625, 1634
\tex_latelua:D	217, 253, 270, 406, 407, 638, 662
\tex_luaescapestring:D	212
\tex luatexversion:D	889
\tex_pdfcompresslevel:D	1633
\tex_pdfdest:D	1439, 1456, 1478, 1486
\tex_pdfextension:D	832, 1508, 1525, 1547, 1555, 1625
\tex_pdflastxform:D	976, 1056
\tex_pdfnames:D	825
\tex_pdfobj:D	1634
\tex_pdfpageattr:D	151
\tex_pdfpagemresources:D	800
\tex_pdfpagesattr:D	114
\tex_pdfrefxform:D	981, 1061
\tex_pdfsuppressptexinfo:D	1600
\tex_pdftexrevision:D	1435
\tex_pdftexversion:D	1434
\tex_pdfvariable:D	1605, 1606
\tex_pdfxform:D	935, 1016
\tex_special:D	133, 281, 320
\tex_the:D	927, 930, 933, 1008, 1011, 1014, 1100, 1103, 1106, 1204, 1207, 1210
\tex_unexpanded:D	212
\tex_vss:D	1386, 1418
text commands:	
\text_expand:n	64, 70
tl commands:	
\c_space_tl	545, 553, 564, 572, 629, 653, 707, 737, 1116, 1117, 1118, 1238
\tl_const:Nn	925, 928, 931, 1006, 1009, 1012, 1098, 1101, 1104, 1202, 1205, 1208
\tl_gput_right:Nn	75, 82, 89
\tl_if_exist:NTF	95
\tl_new:N	49, 1185, 1186, 1187, 1316
\tl_set:Nn	177, 719, 749, 1212, 1217, 1222
\tl_to_str:n	926, 929, 932, 975, 982, 988, 1007, 1010, 1013, 1055, 1063, 1069, 1090, 1099, 1102, 1105, 1160, 1192, 1203, 1206, 1209, 1265, 1289, 1300, 1308, 1459, 1489, 1528, 1558
\tl_use:N	1236, 1241, 1242, 1243
V	
vbox commands:	
\vbox_to_zero:n	1378, 1389