

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

The L^AT_EX Project*

Version 0.96q, released 2025-03-26

1 l3backend-testphase Implementation

```
1 <drivers>\ProvidesExplFile
2 <*dvipdfmx>
3   {l3backend-testphase-dvipdfmx.def}{2025-03-26}{}
4   {LaTeX-PDF-management-testphase-bundle-backend-support: dvipdfmx}
5 </dvipdfmx>
6 <*dvips>
7   {l3backend-testphase-dvips.def}{2025-03-26}{}
8   {LaTeX-PDF-management-testphase-bundle-backend-support: dvips}
9 </dvips>
10 <*dvisvgm>
11   {l3backend-testphase-dvisvgm.def}{2025-03-26}{}
12   {LaTeX-PDF-management-testphase-bundle-backend-support: dvisvgm}
13 </dvisvgm>
14 <*luatex>
15   {l3backend-testphase-luatex.def}{2025-03-26}{}
16   {LaTeX-PDF-management-testphase-bundle-backend-support: PDF output (LuaTeX)}
17 </luatex>
18 <*pdftex>
19   {l3backend-testphase-pdftex.def}{2025-03-26}{}
20   {LaTeX-PDF-management-testphase-bundle-backend-support: PDF output (pdfTeX)}
21 </pdftex>
22 <*xdvipdfmx>
23   {l3backend-testphase-xetex.def}{2025-03-26}{}
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

Commands to get a reference for the absolute page counter.

```

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

```

avoid that destinations names are optimized with xelatex/dvipdfmx see <https://tug.org/pipermail/dvipdfmx/200002.html>

```

82 <*dvipdfmx | xdvipdfmx>
83     \_kernel_backend_literal:n { dvipdfmx:config~C~ 0x0010 }
84 </dvipdfmx | xdvipdfmx>

```

```

\_pdf_backend_resourceid_int
\_pdf_backend_name_int
\_pdf_backend_page_int

```

Some scratch variables

```

85 <*drivers>
86 \prop_new:N \_pdf_backend_resourceid_int
87 \tl_new:N \_pdf_backend_name_int
88 \box_new:N \_pdf_backend_tmpa_box
89 \box_new:N \_pdf_backend_tmpb_box
90 </drivers>

```

(End of definition for _pdf_backend_resourceid_int, _pdf_backend_name_int, and _pdf_backend_tmpa_box.)

```

\_pdf_backend_resourceid_int
\_pdf_backend_name_int
\_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.

```

91 <*drivers>
92 \int_new:N \_pdf_backend_resourceid_int
93 \int_new:N \_pdf_backend_name_int
94 \int_new:N \_pdf_backend_page_int
95 </drivers>

```

(End of definition for _pdf_backend_resourceid_int, _pdf_backend_name_int, and _pdf_backend_page_int.)

1.4 luacode

Load the lua code.

```
96 <*luatex>
97   \directlua { require("l3backend-testphase.lua") }
98 </luatex>
```

1.5 Converting unicode strings to a pdfname

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

```
99 <*pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
100 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
101   {
102     / \str_convert_pdfname:e { \text_expand:n { #1 } }
103   }
104 </pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm>
105 <*dvips>
106 \cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
107   {
108     ~ ( \text_expand:n { #1 } ) ~ cvn
109   }
110 </dvips>
```

1.6 Hooks

1.6.1 Add the “end run” hooks

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

```
111 <*pdftex | luatex>
112 % put in \@kernel@after@enddocument@afterlastpage
113 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
114   {
115     \g__kernel_pdfmanagement_end_run_code_tl
116   }
117 </pdftex | luatex>
118 <*dvipdfmx | xdvipdfmx>
119 % put in \@kernel@after@shipout@lastpage
120 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
121   {
122     \g__kernel_pdfmanagement_end_run_code_tl
123   }
124 </dvipdfmx | xdvipdfmx>
125 <*dvips>
126 % put in \@kernel@after@shipout@lastpage
127 \tl_gput_right:Nn \@kernel@after@shipout@lastpage
128   {
129     \g__kernel_pdfmanagement_end_run_code_tl
130   }
131 </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 ...

```
132 <*drivers>
133 \tl_if_exist:NTF \@kernel@after@shipout@background
134 {
135   \g@addto@macro \@kernel@before@shipout@background{\relax}
136   \g@addto@macro \@kernel@after@shipout@background
137   {
138     \g__kernel_pdfmanagement_thispage_shipout_code_tl
139   }
140 }
141 {
142   \hook_gput_code:nnn{shipout/background}{pdf}
143   {
144     \g__kernel_pdfmanagement_thispage_shipout_code_tl
145   }
146 }
147
148 </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 dviPDFM are additive. luatex sets it in lua. The higher level code has to take this into account.

```
149 <*pdftex>
150 \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
151 {
152   \tex_global:D \tex_pdfpagesattr:D { #1 }
153 }
154 </pdftex>
155 <*luatex>
156 %luatex: does it in lua
157 \sys_if_engine_luatex:T
158 {
159   \cs_new_protected:Npn \_pdf_backend_Pages_primitive:n #1
160   {
161     \tex_directlua:D
162     {
163       pdf.setpagesattributes( \_pdf_backend_luastring:n { #1 } )
164     }
165   }
166 }
167 </luatex>
168 <*dvips>
169 \cs_new_protected:Npx \_pdf_backend_Pages_primitive:n #1
170 {
171   \tex_special:D{ps:~[#1~/PAGES-pdfmark] %}
172 }
```

```

173 </dvips>
174 <*dvipdfmx | xdvipdfmx>
175 \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
176   {
177     \__pdf_backend:n{put~@pages~<<#1>>}
178   }
179 </dvipdfmx | xdvipdfmx>
180 <*dvisvgm>
181 \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
182   {}
183 </dvisvgm>

```

(End of definition for __pdf_backend_Pages_primitive:n.)

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

<pre> __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 </pre>	<p>__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.</p>
--	--

```

184 % backend commands
185 <*pdftex>
186 %the primitive
187 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
188   {
189     \tex_global:D \tex_pdfpageattr:D { #1 }
190   }
191 % the command to store default values.
192 % Uses a prop with pdflatex + dvi,
193 % sets a lua table with luatex
194 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2 %key,value
195   {
196     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
197   }
198 % the command to remove a default value.
199 % Uses a prop with pdflatex + dvi,
200 % changes a lua table with luatex
201 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
202   {
203     \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
204   }
205 % the command used in the document.
206 % direct call of the primitive special with dvips/dvipdfmx
207 % \latelua: fill a page related table with luatex, merge it with the page
208 % table and push it directly
209 % write to aux and store in prop with pdflatex
210 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
211   {
212     %we need to know the page the resource should be added too.

```

```

213 \int_gincr:N\g__pdf_backend_resourceid_int
214 \__pdf_backend_record_abbrev:e { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
215 \tl_set:Ne \l__pdf_tmpa_tl
216 {
217   \__pdf_backend_ref_abbrev:e {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
218 }
219 \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
220 {
221   \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
222 }
223 %backend_Page has no handler.
224 \pdfdict_gput:nnn {g__pdf_Core/backend_Page\l__pdf_tmpa_tl}{ #1 }{ #2 }
225 }
226 %the code to push the values, used in shipout
227 %merges the two props and then fills the register in pdflatex
228 %merges the two tables and then fills (in lua) in luatex
229 %issues the values stored in the global prop with dvi
230 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
231 {
232   \prop_gset_eq:Nc \g__pdf_tmpa_prop { \__kernel_pdfdict_name:n { g__pdf_Core/Page } }
233   \prop_if_exist:cT { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
234   {
235     \prop_map_inline:cn { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
236     {
237       \prop_gput:Nnn \g__pdf_tmpa_prop { ##1 }{ ##2 }
238     }
239   }
240   \__pdf_backend_Page_primitive:e
241   {
242     \prop_map_function:NN \g__pdf_tmpa_prop \pdfdict_item:ne
243   }
244 }
245 </pdfTeX>
246 <*luatex>
247 % do we need to use some escaping for the values?????
248 \cs_new:Npn \__pdf_backend_luastring:n #1
249 {
250   "\tex_luaescapestring:D { \tex_unexpanded:D { #1 } }"
251 }
252 %not used, only there for consistency
253 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
254 {
255   \tex_latelua:D
256   {
257     pdf.setpageattributes(\__pdf_backend_luastring:n { #1 })
258   }
259 }
260 % the command to store default values.
261 % Uses a prop with pdflatex + dvi,
262 % sets a lua table with luatex
263 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
264 {
265   \tex_directlua:D
266   {

```

```

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

```



```

321 % the command to store default values.
322 % Uses a prop with pdflatex + dvi,
323 % sets a lua table with luatex
324 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
325 {
326   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
327 }
328 % the command to remove a default value.
329 % Uses a prop with pdflatex + dvi,
330 % changes a lua table with luatex
331 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
332 {
333   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
334 }
335 % the command used in the document.
336 % direct call of the primitive special with dvips/dvipdfmx
337 % \lattelua: fill a page related table with luatex, merge it with the page
338 % table and push it directly
339 % write to aux and store in prop with pdflatex
340 \cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
341 {
342   \__pdf_backend_Page_primitive:n { /#1~#2 }
343 }
344 %the code to push the values, used in shipout
345 %merges the two props and then fills the register in pdflatex
346 %merges the two tables (the one is probably still empty)
347 % and then fills (in lua) in luatex
348 %issues the values stored in the global prop with dvi
349 \cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
350 {
351   \__pdf_backend_Page_primitive:e
352   { \pdfdict_use:n { g__pdf_Core/Page} }
353 }
354 </dvipdfmx | xdvipdfmx>
355 <*dvips>
356 \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
357 {
358   \tex_special:D{ps:-[ThisPage]<<#1>>~/PUT~pdfmark} %]
359 }
360 % the command to store default values.
361 % Uses a prop with pdflatex + dvi,
362 % sets a lua table with luatex
363 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
364 {
365   \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
366 }
367 % the command to remove a default value.
368 % Uses a prop with pdflatex + dvi,
369 % changes a lua table with luatex
370 \cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
371 {
372   \pdfdict_gremove:nn {g__pdf_Core/Page}{ #1 }
373 }
374 % the command used in the document.

```

```

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

```

419 <*drivers>
420 \clist_const:Nn \c__pdf_backend_PageResources_clist
421 {
422   ExtGState,
423   ColorSpace,
424   Pattern,
425   Shading,
426 }
427 </drivers>

```

(End of definition for `\c__pdf_backend_PageResources_clist`.)

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

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

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

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

`_pdf_backend_PageResources_obj_gpush:`

```

428 <*pdfTeX | luatex>
429 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
430 {
431   \pdf_object_new:n {\_pdf/Page/Resources/#1}
432   \cs_if_exist:NT \tex_directlua:D
433     {
434       \tex_directlua:D
435         {
436           ltx.__pdf.object["\_pdf/Page/Resources/#1"]
437           =
438           "\pdf_object_ref:n{\_pdf/Page/Resources/#1}"
439         }
440     }
441 }
442 </pdfTeX | luatex>

```

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

```

443 <*luatex>
444 \cs_new_protected:Npn \_pdf_backend_PageResources_gput:nnn #1 #2 #3
445 {
446   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
447   \tex_directlua:D{ltx.__pdf.Page.Resources.#1=true}
448   \tex_directlua:D
449     {
450       ltx.pdf.Page_Resources_gpush(tex.count["g_shipout_readonly_int"])
451     }
452 }
453 </luatex>
454 <*pdfTeX>
455 \cs_new_protected:Npn \_pdf_backend_PageResources_gput:nnn #1 #2 #3
456 {

```

```

457     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
458   }
459 </pdfTeX>

```

code for end of document code

```

460 <*pdfTeX | luatex>
461 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush:
462 {
463   \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
464     {
465     \prop_if_empty:cF
466     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
467     {
468     \pdf_object_write:mne
469     { __pdf/Page/Resources/##1 } { dict }
470     { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1} }
471     }
472   }
473 }
474 </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.

```

475 <*dvipdfmx | xdvipdfmx>
476 <xdvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-xetex}{after}{pdf}
477 <dvipdfmx> \hook_gset_rule:nnnn{shipout/firstpage}{l3backend-dvipdfmx}{after}{pdf}
478 %
479 \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
480 {
481   \pdf_object_new:n { __pdf/Page/Resources/#1 }
482   \hook_gput_code:nnn
483     {shipout/firstpage}
484     {pdf}
485     {\pdf_object_write:nnn { __pdf/Page/Resources/#1 } { dict } {}}
486 }
487 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1
488 {
489   \__pdf_backend:n {put~@resources~<<#1>>}
490 }
491 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
492 {
493   % this is not used for output, but there is a test if the resource is empty
494   \prop_gput:cne { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
495     { \str_convert_pdfname:n {#2} }{ #3 }
496   %objects are not filled with \pdf_object_write as this is not additive!
497   \__pdf_backend:e
498   {
499     put~\pdf_object_ref:n {__pdf/Page/Resources/#1}<</#2~#3>>
500   }
501 }
502
503 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}

```

```
504 </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.

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

dvipsvgm unneeded, or no-op

```
513 <*dvisvgm>
514 \cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
515 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
516 { %only for the show command TEST!!
517   \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
518 }
519 \cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
520 </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 the backend command that create the bdc/emc marker and store the properties.
\__pdf_backend_bdcobject:n \__pdf_backend_PageResources_gpush:n outputs the /Properties and/or the other re-
  \__pdf_backend_bmc:n sources for the current page.
  \__pdf_backend_emc: pdftex and luatex (and perhaps dvips ...) need to know if there are in a xform stream
\__pdf_backend_PageResources_gpush:n ...
```

```
521 <*drivers>
522 \bool_new:N \l__pdf_backend_xform_bool
523 </drivers>
```

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

```
524 <*dvips>
525 %
526 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
527 {
528   \__pdf_backend_pdfmark:n{/#1~<<#2>>~/BDC}
529 }
```

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

```
530 \cs_set_eq:NN \__pdf_backend_bdc_contobj:nn \__pdf_backend_bdc:nn
531 \cs_set_eq:NN \__pdf_backend_bdc_contstream:nn \__pdf_backend_bdc:nn
532
533 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
```

```

534 {
535   \cs_new_protected:Npn \__pdf_backend_bdc_shipout:ee #1 #2 % #1 eg. Span, #2: dict_content
536   {
537     \__kernel_backend_shipout_literal:e
538     {ps: SDict ~ begin ~ mark /#1~<<#2>>~/BDC ~ pdfmark ~ end }
539   }
540 }
541
542 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
543 {
544   \__pdf_backend_pdfmark:e{/#1~\pdf_object_ref:n{#2}~/BDC}
545 }
546 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
547 {
548   \__pdf_backend_pdfmark:e{/#1~\__pdf_backend_object_last:~/BDC}
549 }
550 \cs_set_protected:Npn \__pdf_backend_emc:
551 {
552   \__pdf_backend_pdfmark:n{/EMC} %
553 }
554 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
555 {
556   \__pdf_backend_pdfmark:n{/#1~/BMC} %
557 }
558 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
559
560 </dvips>
561 <*dvisvgm>
562 % dvisvgm should do nothing
563 %
564 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
565 {}
566 \cs_set_eq:NN \__pdf_backend_bdc_contobj:nn \__pdf_backend_bdc:nn
567 \cs_set_eq:NN \__pdf_backend_bdc_contstream:nn \__pdf_backend_bdc:nn
568
569 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
570 {
571   \cs_set_protected:Npn \__pdf_backend_shipout_bdc:ee #1 #2 % #1 eg. Span, #2: dict_content
572   {}
573 }
574 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
575 {}
576 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span,
577 {}
578 \cs_set_protected:Npn \__pdf_backend_emc:
579 {}
580 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
581 {}
582 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
583
584 </dvisvgm>
585 %
586 % xetex has to create the entries in the /Properties manually
587 % (like the other backends)

```

```

588 % use pdfbase special
589 % https://chat.stackexchange.com/transcript/message/50832016#50832016
590 % the property is added to xform resources automatically,
591 % no need to worry about it.
592 (*dvipdfmx | xdvipdfmx)
593 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
594 {
595   \int_gincr:N \g__pdf_backend_name_int
596   \__kernel_backend_literal:e
597   {
598     pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
599   }
600   \__kernel_backend_literal:e
601   {
602     pdf:put~@resources~
603     <<
604       /Properties~
605       <<
606         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
607         \pdf_object_ref:n { #2 }
608       >>
609     >>
610   }
611 }
612 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1 % #1 eg. Span
613 {
614   \int_gincr:N \g__pdf_backend_name_int
615   \__kernel_backend_literal:e
616   {
617     pdf:code~/\exp_not:n{#1}/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
618   }
619   \__kernel_backend_literal:e
620   {
621     pdf:put~@resources~
622     <<
623       /Properties~
624       <<
625         /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
626         \__pdf_backend_object_last:
627       >>
628     >>
629   }
630 }
631 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
632 {
633   \__kernel_backend_literal:n {pdf:code~/#1~BMC} %pdfbase
634 }
635
636 %this require management
637 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
638 {
639   \pdf_object_unnamed_write:nn { dict }{ #2 }
640   \__pdf_backend_bdcobject:n { #1 }
641 }

```

```

642
643 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
644 {
645   \__kernel_backend_literal:n {pdf:code~ /#1~<<#2>>~BDC }
646 }
647
648 \cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
649 {
650   \bool_if:NTF \g_pdfmanagement_active_bool
651     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
652     {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
653     \__pdf_backend_bdc:nn {#1}{#2}
654 }
655
656 \bool_if:NT\l_pdfmanagement_delayed_shipout_bool
657 {
658   \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
659   {
660     \__kernel_backend_shipout_literal:e {pdf:code~ /#1~<<#2>>~BDC }
661   }
662   \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
663 }
664 \cs_set_protected:Npn \__pdf_backend_emc:
665 {
666   \__kernel_backend_literal:n {pdf:code~EMC} %pdfbase
667 }
668 % properties are handled automatically, but the other resources should be added
669 % at shipout
670 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
671 {
672   \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
673   {
674     \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
675     {
676       \__kernel_backend_literal:e
677       {
678         pdf:put~@resources~
679         <</#1~\pdf_object_ref:n {__pdf/Page/Resources/##1}>>
680       }
681     }
682   }
683 }
684 </dvipdfmx | xdvipdfmx>
685 % luatex + pdftex
686 <*luatex>
687 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
688 {
689   \int_gincr:N \g_pdf_backend_name_int
690   \__kernel_backend_literal_page:e
691   { /#1 ~ /l3pdf\int_use:N\g_pdf_backend_name_int\c_space_tl BDC }
692   \bool_if:NTF \l_pdf_backend_xform_bool
693   {
694     \pdfdict_gput:nee
695     { g__pdf_Core/Xform/Resources/Properties }

```



```

696         { l3pdf\int_use:N\g__pdf_backend_name_int }
697         { \pdf_object_ref:n { #2 } }
698     }
699     {
700     \exp_args:Ne \tex_latelua:D
701     {
702         ltx.pdf.Page_Resources_Properties_gput
703         (
704             tex.count["g_shipout_readonly_int"],
705             "l3pdf\int_use:N\g__pdf_backend_name_int",
706             "\pdf_object_ref:n { #2 }"
707         )
708     }
709 }
710 }
711 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
712 {
713     \int_gincr:N \g__pdf_backend_name_int
714     \__kernel_backend_literal_page:e
715     { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
716     \bool_if:NTF \l__pdf_backend_xform_bool
717     {
718         \pdfdict_gput:nee %no handler needed
719         { g__pdf_Core/Xform/Resources/Properties }
720         { l3pdf\int_use:N\g__pdf_backend_name_int }
721         { \__pdf_backend_object_last: }
722     }
723     {
724     \exp_args:Ne \tex_latelua:D
725     {
726         ltx.pdf.Page_Resources_Properties_gput
727         (
728             tex.count["g_shipout_readonly_int"],
729             "l3pdf\int_use:N\g__pdf_backend_name_int",
730             "\__pdf_backend_object_last:"
731         )
732     }
733 }
734 }
735 \cs_set_protected:Npn \__pdf_backend_bmc:n #1
736 {
737     \__kernel_backend_literal_page:n { /#1~BMC }
738 }
739 \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
740 {
741     \pdf_object_unnamed_write:nn { dict } { #2 }
742     \__pdf_backend_bdcobject:n { #1 }
743 }
744 \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
745 {
746     \__kernel_backend_literal_page:n { /#1-<<#2>>~BDC }
747 }
748
749 \cs_set_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn

```

```

750
751 \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
752 {
753   \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
754   {
755     \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
756   }
757   \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
758 }
759
760 \cs_set_protected:Npn \__pdf_backend_emc:
761 {
762   \__kernel_backend_literal_page:n { EMC }
763 }
764
765 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
766 </luatex>

```

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

```

767 <*pdfTeX>
768 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
769 {
770   \int_gincr:N \g__pdf_backend_name_int
771   \__kernel_backend_literal_page:e
772   { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
773   % code to set the property ...
774   \int_gincr:N\g__pdf_backend_resourceid_int
775   \bool_if:NTF \l__pdf_backend_xform_bool
776   {
777     \pdfdict_gput:nee %no handler needed
778     { g__pdf_Core/Xform/Resources/Properties }
779     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
780     { \pdf_object_ref:n { #2 } }
781   }
782   {
783     \__pdf_backend_record_abspage:e {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
784     \tl_set:Ne \l__pdf_tmpa_tl
785     {
786       \__pdf_backend_ref_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
787     }
788     \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
789     {
790       \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
791     }
792     \pdfdict_gput:nee
793     { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
794     { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
795     { \pdf_object_ref:n{#2} }
796   }
797 }
798 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
799 {
800   \int_gincr:N \g__pdf_backend_name_int

```

```

801  \__kernel_backend_literal_page:e
802  { /\exp_not:n{#1} ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
803  % code to set the property ...
804  \int_gincr:N\g__pdf_backend_resourceid_int
805  \bool_if:NTF \l__pdf_backend_xform_bool
806  {
807    \pdfdict_gput:nee
808    { g__pdf_Core/Xform/Resources/Properties }
809    { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
810    { \__pdf_backend_object_last: }
811  }
812  {
813    \__pdf_backend_record_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
814    \tl_set:Ne \l__pdf_tmpa_tl
815    {
816      \__pdf_backend_ref_abspage:e{l3pdf\int_use:N\g__pdf_backend_resourceid_int}
817    }
818    \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
819    {
820      \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
821    }
822    \pdfdict_gput:nee
823    { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
824    { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
825    { \__pdf_backend_object_last: }
826    %\pdfdict_show:n { g_backend_Page\l__pdf_tmpa_tl/Resources/Properties }
827  }
828  }
829  \cs_set_protected:Npn \__pdf_backend_bmc:n #1
830  {
831    \__kernel_backend_literal_page:n { /#1~BMC }
832  }
833  \cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
834  {
835    \pdf_object_unnamed_write:nn { dict } { #2 }
836    \__pdf_backend_bdcobject:n { #1 }
837  }
838  \cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
839  {
840    \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
841  }

```

We use by default the direct BDC.

```

842  \cs_set_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn
843
844  \bool_if:NT\l__pdfmanagement_delayed_shipout_bool
845  {
846    \cs_set_protected:Npn \__pdf_backend_bdc_shipout_contstream:ee #1 #2
847    {
848      \__kernel_backend_shipout_literal_page:e { /#1~<<#2>>~BDC }
849    }
850    \cs_set_eq:NN \__pdf_backend_bdc_shipout:ee \__pdf_backend_bdc_shipout_contstream:ee
851  }
852
853  \cs_set_protected:Npn \__pdf_backend_emc:

```

```

854 {
855   \__kernel_backend_literal_page:n { EMC }
856 }
857
858 \cs_new:Npn \__pdf_backend_PageResources_gpush_aux:n #1 %#1 ExtGState etc
859 {
860   \prop_if_empty:cF
861     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
862     {
863       \pdfdict_item:ne { #1 }{ \pdf_object_ref:n {__pdf/Page/Resources/#1}}
864     }
865 }
866
867 \cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
868 {
869   \exp_args:NNe \tex_global:D \tex_pdfpageresources:D
870   {
871     \prop_if_exist:cT
872       { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
873       {
874         /Properties~
875         <<
876         \prop_map_function:cN
877           { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Property
878             \pdfdict_item:ne
879           }
880         }
881         %% add ExtGState etc
882         \clist_map_function:NN
883           \c__pdf_backend_PageResources_clist
884           \__pdf_backend_PageResources_gpush_aux:n
885       }
886   }
887
888 </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.

```

889 % pdflatex
890 <*pdftex>
891 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
892 {
893   \pdf_object_unnamed_write:nn {dict} {/Names [#2] }

```

```

894     \tex_pdfnames:D {/#1~\pdf_object_ref_last:}
895   }
896 </pdftex>
897 <*luatex>
898 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
899   {
900     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
901     \tex_pdfextension:D~names~ {/#1~\pdf_object_ref_last:}
902   }
903 </luatex>
904 <*dvipdfmx | xdvipdfmx>
905 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
906   {
907     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
908     \__pdf_backend:e {put~@names~<</#1~\pdf_object_ref_last: >>}
909   }
910 </dvipdfmx | xdvipdfmx>
911
912 %dvips: noop
913 <*dvips>
914 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
915 </dvips>
916 %dvisvgm: noop
917 <*dvisvgm>
918 \cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
919 </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.

```

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

```
940 <*drivers>
941 \cs_if_free:NT \__pdf_backend_link_off:
942 {
943   \cs_new_protected:Npn \__pdf_backend_link_off: {}
944   \cs_new_protected:Npn \__pdf_backend_link_on: {}
945 }
946 </drivers>
947 <*pdftex>
948 \cs_if_exist:NT \pdfrunninglinkoff
949 {
950   \cs_set_protected:Npn \__pdf_backend_link_off:
951     {
952       \pdfrunninglinkoff
953     }
954   \cs_set_protected:Npn \__pdf_backend_link_on:
955     {
956       \pdfrunninglinkon
957     }
958 }
959 </pdftex>
960 <*luatex>
961 \int_compare:nNnT {\tex_luatexversion:D } > {112}
962 {
963   \cs_set_protected:Npn \__pdf_backend_link_off:
964     {
965       \pdfextension linkstate 1
966     }
967   \cs_set_protected:Npn \__pdf_backend_link_on:
968     {
969       \pdfextension linkstate 0
970     }
971 }
972 </luatex>
973 <*dvipdfmx | xdvipdfmx>
974 \cs_set_protected:Npn \__pdf_backend_link_off:
975   {
976     \__pdf_backend:n { nolinek }
977   }
978 \cs_set_protected:Npn \__pdf_backend_link_on:
979   {
980     \__pdf_backend:n { link }
981   }
982 </dvipdfmx | xdvipdfmx>
```

1.10.3 Form XObject / backend

```
\__pdf_backend_xform_new:nnnn #1 : name
                               #2 : attributes
                               #3 : resources needed?? or are all resources autogenerated?
```

#4 : content, this doesn't need to be a box!

```
\__pdf_backend_xform_use:n 983 <*pdfTeX>
\__pdf_backend_xform_ref:n 984 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
985 % #1 name
986 % #2 attributes
987 % #3 resources
988 % #4 content, not necessarily a box!
989 {
990   \hbox_set:Nn \l__pdf_backend_tmpa_box
991   {
992     \bool_set_true:N \l__pdf_backend_xform_bool
993     \prop_gc_clear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
994     #4
995   }
996   %store the dimensions
997   \tl_const:ce
998   { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
999   { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1000  \tl_const:ce
1001  { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1002  { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1003  \tl_const:ce
1004  { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1005  { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1006  %% do we need to test if #2 and #3 are empty??
1007  \tex_immediate:D \tex_pdfxform:D
1008  ~ attr ~ { #2 }
1009  %% which other resources should be default? Is an argument actually needed?
1010  ~ resources ~
1011  {
1012    #3
1013    \int_compare:nNnT
1014    { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1015    >
1016    { 0 }
1017    {
1018      /Properties~
1019      <<
1020      \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1021      >>
1022    }
1023  }
1024  \prop_if_empty:cF
1025  { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1026  {
1027    /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1028  }
1029  \prop_if_empty:cF
1030  { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1031  {
1032    /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1033  }
```

```

1034     \prop_if_empty:cF
1035     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1036     {
1037         /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1038     }
1039     \prop_if_empty:cF
1040     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1041     {
1042         /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1043     }
1044 }
1045 \l__pdf_backend_tmpa_box
1046 \int_const:cn
1047 { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1048 { \tex_pdflastxform:D }
1049 }
1050
1051 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1052 {
1053     \tex_pdfrefxform:D
1054     \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1055     \scan_stop:
1056 }
1057
1058 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1059 {
1060     \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
1061 }
1062 </pdftex>
1063 <*luatex>
1064 %luatex
1065 %nearly identical but not completely ...
1066 \cs_new_protected:Npn \__pdf_backend_xform_new:n #1 #2 #3 #4
1067 % #1 name
1068 % #2 attributes
1069 % #3 resources
1070 % #4 content, not necessarily a box!
1071 {
1072     \hbox_set:Nn \l__pdf_backend_tmpa_box
1073     {
1074         \bool_set_true:N \l__pdf_backend_xform_bool
1075         \prop_gc_clear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1076         #4
1077     }
1078     \tl_const:ce
1079     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1080     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1081     \tl_const:ce
1082     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1083     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1084     \tl_const:ce
1085     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1086     { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1087     %% do we need to test if #2 and #3 are empty??

```



```

1088 \tex_immediate:D \tex_pdfxform:D
1089 ~ attr ~ { #2 }
1090 %% which resources should be default? Is an argument actually needed?
1091 ~ resources ~
1092 {
1093   #3
1094   \int_compare:nNnT
1095     {\prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties
1096     >
1097     { 0 }
1098     {
1099       /Properties~
1100       <<
1101       \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
1102       >>
1103     }
1104   \prop_if_empty:cF
1105     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
1106     {
1107       /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1108     }
1109   \prop_if_empty:cF
1110     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
1111     {
1112       /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1113     }
1114   \prop_if_empty:cF
1115     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
1116     {
1117       /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1118     }
1119   \prop_if_empty:cF
1120     { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
1121     {
1122       /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1123     }
1124   }
1125   \l__pdf_backend_tmpa_box
1126   \int_const:cn
1127     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1128     { \tex_pdflastxform:D }
1129 }
1130
1131 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 %protected as with xelatex
1132 {
1133   \tex_pdfrefxform:D \int_use:c
1134   {
1135     c__pdf_backend_xform_ \tl_to_str:n {#1} _int
1136   }
1137   \scan_stop:
1138 }
1139
1140 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1141 { \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

```

```

1142
1143 </luatex>
1144 <{*dvi.pdfmx | xdvipdfmx}>
1145 % xetex
1146 % it needs a bit testing if it really works to set the box to 0 before the special ...
1147 % does it disturb viewing the xobject?
1148 % what happens with the resources (bdc)? (should work as they are specials too)
1149 % xetex requires that the special is in horizontal mode. This means it affects
1150 % typesetting. But we can no delay the whole form code to shipout
1151 % as the object reference and the size is often wanted on the current page.
1152 % so we need to allocate a box - but probably they won't be thousands xform
1153 % in a document so it shouldn't matter.
1154 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
1155 % #1 name
1156 % #2 attributes
1157 % #3 resources
1158 % #4 content, not necessarily a box!
1159 {
1160   \int_gincr:N \g__pdf_backend_object_int
1161   \int_const:cn
1162     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1163   { \g__pdf_backend_object_int }
1164   \box_new:c { g__pdf_backend_xform_#1_box }
1165   \hbox_gset:cn { g__pdf_backend_xform_#1_box }
1166   {
1167     \bool_set_true:N \l__pdf_backend_xform_bool
1168     #4
1169   }
1170   \tl_const:ce
1171     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1172     { \tex_the:D \box_wd:c { g__pdf_backend_xform_#1_box } }
1173   \tl_const:ce
1174     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1175     { \tex_the:D \box_ht:c { g__pdf_backend_xform_#1_box } }
1176   \tl_const:ce
1177     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1178     { \tex_the:D \box_dp:c { g__pdf_backend_xform_#1_box } }
1179   \box_set_dp:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1180   \box_set_ht:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1181   \box_set_wd:cn { g__pdf_backend_xform_#1_box } { \c_zero_dim }
1182   \hook_gput_next_code:nm {shipout/background}
1183   {
1184     \mode_leave_vertical: %needed, the xform disappears without it.
1185     \__pdf_backend:e
1186     {
1187       bobj ~ \__pdf_backend_xform_ref:n { #1 }
1188       \c_space_tl width ~ \pdfxform_wd:n { #1 }
1189       \c_space_tl height ~ \pdfxform_ht:n { #1 }
1190       \c_space_tl depth ~ \pdfxform_dp:n { #1 }
1191     }
1192     \box_use_drop:c { g__pdf_backend_xform_#1_box }
1193     \__pdf_backend:e {put ~ @resources ~<<#3>> }
1194     \__pdf_backend:e
1195     {

```

```

1196         put~ @resources ~
1197         <<
1198         /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
1199         >>
1200     }
1201     \__pdf_backend:e
1202     {
1203         put~ @resources ~
1204         <<
1205         /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
1206         >>
1207     }
1208     \__pdf_backend:e
1209     {
1210         put~ @resources ~
1211         <<
1212         /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
1213         >>
1214     }
1215     \__pdf_backend:e
1216     {
1217         put~ @resources ~
1218         <<
1219         /ColorSpace~
1220         \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
1221         >>
1222     }
1223     \__pdf_backend:e {exobj ~<<#2>>}
1224 }
1225 }
1226
1227
1228
1229 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1230 {
1231     @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1232 }
1233
1234 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1235 {
1236     \hbox_set:Nn \l__pdf_backend_tmpa_box
1237     {
1238         \__pdf_backend:e
1239         {
1240             uxobj~ \__pdf_backend_xform_ref:n { #1 }
1241         }
1242     }
1243     \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1244     \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1245     \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1246     \box_use_drop:N \l__pdf_backend_tmpa_box
1247 }
1248 </dviptfm | xdvipdfmx>
1249 <*dvisvgm>

```

```

1250 % unclear what it should do!!
1251 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}
1252 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
1253 \cs_new:Npn \__pdf_backend_xform_ref:n {}
1254 </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

```

1255 <*dvips>
1256 \tl_new:N \l__pdf_backend_xform_tmpwd_tl
1257 \tl_new:N \l__pdf_backend_xform_tmpdp_tl
1258 \tl_new:N \l__pdf_backend_xform_tmph_tl

1259 \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 % #1 name, #2 attribute, #4
1260 {
1261   \int_gincr:N \g__pdf_backend_object_int
1262   \int_const:cn
1263     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1264     { \g__pdf_backend_object_int }
1265
1266   \hbox_set:Nn \l__pdf_backend_tmpa_box
1267     {
1268       \bool_set_true:N \l__pdf_backend_xform_bool
1269       \prop_gc clear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }
1270       #4
1271     }
1272   %store the dimensions
1273   \tl_const:ce
1274     { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
1275     { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
1276   \tl_const:ce
1277     { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
1278     { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
1279   \tl_const:ce
1280     { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
1281     { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
1282   %store content dimensions in DPI units (Dots) (code from issue 25)
1283   \tl_set:Nc \l__pdf_backend_xform_tmpwd_tl
1284     {
1285       \dim_to_decimal_in_sp:n{ \box_wd:N \l__pdf_backend_tmpa_box }~
1286       65536~div~72.27~div~DVImag~mul~Resolution~mul~
1287     }
1288   \tl_set:Nc \l__pdf_backend_xform_tmph_tl
1289     {
1290       \dim_to_decimal_in_sp:n{ \box_ht:N \l__pdf_backend_tmpa_box }~
1291       65536~div~72.27~div~DVImag~mul~VResolution~mul~
1292     }
1293   \tl_set:Nc \l__pdf_backend_xform_tmpdp_tl
1294     {
1295       \dim_to_decimal_in_sp:n{ \box_dp:N \l__pdf_backend_tmpa_box }~
1296       65536~div~72.27~div~DVImag~mul~VResolution~mul~
1297     }
1298   % mirror the box

```

```

1299 %\box_scale:Nnn \l__pdf_backend_tmpa_box {1} {-1}
1300 \hbox_set:Nn\l__pdf_backend_tmpb_box
1301 {
1302   \__kernel_backend_postscript:e
1303   {
1304     gsave~currentpoint~
1305     initclip~ % restore default clipping path (page device/whole page)
1306     clippath~pathbbox~newpath~pop~pop~
1307     \tl_use:N\l__pdf_backend_xform_tmpdp_tl~add~translate~
1308     mark~
1309     /_objdef~{ pdf.obj \int_use:N\g__pdf_backend_object_int }\c_space_tl~
1310     /BBox[
1311       0~
1312       \tl_use:N\l__pdf_backend_xform_tmpht_tl~
1313       \tl_use:N\l__pdf_backend_xform_tmpwd_tl~
1314       \tl_use:N\l__pdf_backend_xform_tmpdp_tl~
1315       neg
1316     ]
1317     \str_if_eq:eeF{#1}{}
1318     {
1319       product~(Distiller)~search~{pop~pop~pop~#2}{pop}ifelse~
1320     }
1321     /BP~pdfmark~1~-1~scale~neg~exch~neg~exch~translate
1322   }
1323   \box_use_drop:N\l__pdf_backend_tmpa_box
1324   \__kernel_backend_postscript:n
1325   {
1326     mark ~ /EP~pdfmark ~ grestore
1327   }
1328   \str_if_eq:eeF{#1}{}
1329   {
1330     \__kernel_backend_postscript:e
1331     {
1332       product~(Ghostscript)~search~
1333       {
1334         pop~pop~pop~
1335         mark~
1336         { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1337         ~<<#2>>~/PUT~pdfmark
1338       }{pop}ifelse
1339     }
1340   }
1341 }
1342 \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1343 \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1344 \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
1345 \hook_gput_code:nnn {begindocument/end}{pdfxform}
1346 {
1347   \mode_leave_vertical:
1348   \box_use:N\l__pdf_backend_tmpb_box
1349 }
1350 }
1351
1352

```

```

1353 \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
1354 {
1355   \hbox_set:Nn \l__pdf_backend_tmpa_box
1356   {
1357     \__kernel_backend_postscript:e
1358     {
1359       gsave~currentpoint~translate~1~-1~scale~
1360       mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int }}~
1361       /SP~pdfmark ~ grestore
1362     }
1363   }
1364   \box_set_wd:Nn \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
1365   \box_set_ht:Nn \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
1366   \box_set_dp:Nn \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
1367   \box_use_drop:N \l__pdf_backend_tmpa_box
1368 }
1369 \cs_new:Npn \__pdf_backend_xform_ref:n #1
1370 {
1371   { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
1372 }
1373
1374 </dvips>
1375 <*drivers>
1376 %% all
1377 \prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
1378 {
1379   \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
1380   { \prg_return_true: }
1381   { \prg_return_false: }
1382 }
1383 \prg_new_eq_conditional:NMn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
1384 { TF , T , F , p }
1385 </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 knew new keywords which allow to create structure destination easily.

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

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

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

```
\tl_set:Nn \l_pdf_current_structure_destination_tl { __tag/struct/\g__tag_struct_stack
```

or if indexed structure object names are used

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

```
1386 <*drivers>
1387 \tl_new:N \l_pdf_current_structure_destination_tl
1388 </drivers>
```

(End of definition for \l_pdf_current_structure_destination_tl.)

We will define alternatives for three backend commands:

```
\__pdf_backend_destination:nn      -> \__pdf_backend_structure_destination:nn
\__pdf_backend_destination:nmnm    -> \__pdf_backend_structure_destination:nmnm
\__pdf_backend_link_begin_goto:nmw -> \__pdf_backend_link_begin_structure_goto:nmw
\__pdf_backend_destination:nn      -> \__pdf_backend_indexed_structure_destination:nn
\__pdf_backend_destination:nmnm    -> \__pdf_backend_indexed_structure_destination:nmnm
\__pdf_backend_link_begin_goto:nmw -> \__pdf_backend_indexed_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:
\pdf_activate_indexed_structure_destination:
1389 <*drivers>
1390 \cs_new_protected:Npn \pdf_activate_structure_destination:
1391 {
1392   \cs_gset_eq:NN \__pdf_backend_destination:nn      \__pdf_backend_structure_destination:nn
1393   \cs_gset_eq:NN \__pdf_backend_destination:nmnm    \__pdf_backend_structure_destination:nmnm
1394   \cs_gset_eq:NN \__pdf_backend_link_begin_goto:nmw \__pdf_backend_link_begin_structure_goto:nmw
1395 }
1396 \cs_new_protected:Npn \pdf_activate_indexed_structure_destination:
1397 {
1398   \cs_gset_eq:NN \__pdf_backend_destination:nn      \__pdf_backend_indexed_structure_destination:nn
1399   \cs_gset_eq:NN \__pdf_backend_destination:nmnm    \__pdf_backend_indexed_structure_destination:nmnm
1400   \cs_gset_eq:NN \__pdf_backend_link_begin_goto:nmw \__pdf_backend_indexed_link_begin_structure_goto:nmw
1401 }
1402 </drivers>
```

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

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

```

1403 <*drivers>
1404 \cs_set_eq:NN \__pdf_backend_structure_destination:nn \__pdf_backend_destination:nn
1405 \cs_set_eq:NN \__pdf_backend_structure_destination:nmnn \__pdf_backend_destination:nmnn
1406 \cs_set_eq:NN \__pdf_backend_link_begin_structure_goto:nw \__pdf_backend_link_begin_goto:nw
1407 \cs_set_eq:NN \__pdf_backend_indexed_structure_destination:nn \__pdf_backend_destination:nn
1408 \cs_set_eq:NN \__pdf_backend_indexed_structure_destination:nmnn \__pdf_backend_destination:nmnn
1409 </drivers>

```

```

\__pdf_backend_structure_destination:nn
\__pdf_backend_structure_destination:nmnn
\__pdf_backend_link_begin_structure_goto:nw

```

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

```

1410 <*xdvipdfmx | dvipdfmx>
1411 \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1412 {
1413   \__pdf_backend:e
1414   {
1415     dest ~ ( \exp_not:n {#1} )
1416     [
1417       @thispage
1418       \str_case:nnF {#2}
1419       {
1420         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1421         { fit } { /Fit }
1422         { fitb } { /FitB }
1423         { fitbh } { /FitBH }
1424         { fitbv } { /FitBV ~ @xpos }
1425         { fith } { /FitH ~ @ypos }
1426         { fitv } { /FitV ~ @xpos }
1427         { fitr } { /Fit }
1428       }
1429       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1430     ]
1431   }

```

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.

```

1432 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1433 {
1434   \__pdf_backend:e
1435   {
1436     obj ~ @pdf.Sdest.\exp_not:n{#1}
1437     [
1438       \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
1439       \str_case:nnF {#2}
1440       {
1441         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1442         { fit } { /Fit }

```



```

1443         { fitb } { /FitB }
1444         { fitbh } { /FitBH }
1445         { fitbv } { /FitBV ~ @xpos }
1446         { fith } { /FitH ~ @ypos }
1447         { fitv } { /FitV ~ @xpos }
1448         { fitr } { /Fit }
1449     }
1450     { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1451 ]
1452 }
1453 }
1454 }

```

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

```

1455 \cs_new_protected:Npn \__pdf_backend_structure_destination_aux:nnnn #1#2#3#4
1456 {
1457     \vbox_to_zero:n
1458     {
1459         \__kernel_kern:n {#4}
1460         \hbox:n
1461         {
1462             \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1463             \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1464         }
1465         \tex_vss:D
1466     }
1467     \__kernel_kern:n {#1}
1468     \vbox_to_zero:n
1469     {
1470         \__kernel_kern:n { -#3 }
1471         \hbox:n
1472         {
1473             \__pdf_backend:n
1474             {
1475                 dest ~ (#2)
1476                 [
1477                     @thispage
1478                     /FitR ~
1479                     @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1480                     @xpos ~ @ypos
1481                 ]
1482             }
1483         }
1484     }

```

Here we add the structure destination to the same box

```

1483     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1484     {
1485         \__pdf_backend:e
1486         {
1487             obj ~ @pdf.SDest.\exp_not:n{#2}
1488             [
1489                 \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_
1490                 /FitR ~
1491                 @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1492                 @xpos ~ @ypos

```

```

1493         ]
1494     }
1495 }
1496 }
1497 \tex_vss:D
1498 }
1499 \__kernel_kern:n { -#1 }
1500 }

```

And now we redefine the destination command:

```

1501 \cs_set_protected:Npn \__pdf_backend_structure_destination:nmnn #1#2#3#4
1502 {
1503   \exp_args:Ne \__pdf_backend_structure_destination_aux:nmnn
1504   { \dim_eval:n {#2} } {#1} {#3} {#4}
1505 }

```

At last the goto link.

```

1506 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nmw #1#2
1507 {
1508   \__pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD~@pdf.SDest.
1509 }
1510 </xdvipdfmx | dvipdfmx>

```

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

```

1511 <*pdftex>
1512 \bool_lazy_and:nnT
1513 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1514 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
1515 {
1516   \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1517   {
1518     \tex_pdfdest:D
1519     name {#1}
1520     \str_case:nnF {#2}
1521     {
1522       { xyz } { xyz }
1523       { fit } { fit }
1524       { fitb } { fitb }
1525       { fitbh } { fitbh }
1526       { fitbv } { fitbv }
1527       { fith } { fith }
1528       { fitv } { fitv }
1529       { fitr } { fitr }
1530     }
1531     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1532     \scan_stop:
1533     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1534     {
1535       \tex_pdfdest:D
1536       struct~
1537       \int_use:c
1538       { c_pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destin
1539       name {#1}
1540       \str_case:nnF {#2}
1541       {

```

```

1542         { xyz } { xyz }
1543         { fit } { fit }
1544         { fitb } { fitb }
1545         { fitbh } { fitbh }
1546         { fitbv } { fitbv }
1547         { fith } { fith }
1548         { fitv } { fitv }
1549         { fitr } { fitr }
1550     }
1551     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1552     \scan_stop:
1553 }
1554 }
1555 \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1556 {
1557     \tex_pdfdest:D
1558     name {#1}
1559     fitr ~
1560     width \dim_eval:n {#2} ~
1561     height \dim_eval:n {#3} ~
1562     depth \dim_eval:n {#4} \scan_stop:
1563     \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1564     {
1565         \tex_pdfdest:D
1566         struct~
1567         \int_use:c
1568         { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destination_tl}
1569         name {#1}
1570         fitr ~
1571         width \dim_eval:n {#2} ~
1572         height \dim_eval:n {#3} ~
1573         depth \dim_eval:n {#4} \scan_stop:
1574     }
1575 }
1576 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1577 {
1578     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1579 }
1580 }
1581 </pdfTeX>

```

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

```

1582 <*luatex>
1583 \int_compare:nNnT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1584 {
1585     \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
1586     {
1587         \tex_pdfextension:D dest
1588         name {#1}
1589         \str_case:nnF {#2}
1590         {
1591             { xyz } { xyz }
1592             { fit } { fit }
1593             { fitb } { fitb }
1594             { fitbh } { fitbh }

```

```

1595         { fitbv } { fitbv }
1596         { fith } { fith }
1597         { fitv } { fitv }
1598         { fitr } { fitr }
1599     }
1600     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1601     \scan_stop:
1602 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1603 {
1604     \tex_pdfextension:D dest
1605     struct~
1606     \int_use:c
1607     { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destin
1608     name {#1}
1609     \str_case:nnF {#2}
1610     {
1611         { xyz } { xyz }
1612         { fit } { fit }
1613         { fitb } { fitb }
1614         { fitbh } { fitbh }
1615         { fitbv } { fitbv }
1616         { fith } { fith }
1617         { fitv } { fitv }
1618         { fitr } { fitr }
1619     }
1620     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1621     \scan_stop:
1622 }
1623 }
1624 \cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
1625 {
1626     \tex_pdfextension:D dest
1627     name {#1}
1628     fitr ~
1629     width \dim_eval:n {#2} ~
1630     height \dim_eval:n {#3} ~
1631     depth \dim_eval:n {#4} \scan_stop:
1632 \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
1633 {
1634     \tex_pdfextension:D dest
1635     struct~
1636     \int_use:c
1637     { c__pdf_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_destinat
1638     name {#1}
1639     fitr ~
1640     width \dim_eval:n {#2} ~
1641     height \dim_eval:n {#3} ~
1642     depth \dim_eval:n {#4} \scan_stop:
1643 }
1644 }
1645 \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1646 {
1647     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1648 }

```

```

1649 }
1650 </luatex>

```

(End of definition for `_pdf_backend_structure_destination:nn`, `_pdf_backend_structure_destination:nmmn`, and `_pdf_backend_link_begin_structure_goto:nmw`.)

```

df_backend_indexed_structure_destination:nn
_backend_indexed_structure_destination:nmmn

```

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

```

1651 <*xdvipdfmx | dvipdfmx>
1652 \cs_set_protected:Npn \_pdf_backend_indexed_structure_destination:nn #1#2
1653 {
1654   \_pdf_backend:e
1655   {
1656     dest ~ ( \exp_not:n {#1} )
1657     [
1658       @thispage
1659       \str_case:nnF {#2}
1660       {
1661         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1662         { fit } { /Fit }
1663         { fitb } { /FitB }
1664         { fitbh } { /FitBH }
1665         { fitbv } { /FitBV ~ @xpos }
1666         { fith } { /FitH ~ @ypos }
1667         { fitv } { /FitV ~ @xpos }
1668         { fitr } { /Fit }
1669       }
1670       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1671     ]
1672   }

```

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

```

1673   \_pdf_backend:e
1674   {
1675     obj ~ @pdf.SDest.\exp_not:n{#1}
1676     [
1677       \exp_after:wN \pdf_object_ref_indexed:nn \l_pdf_current_structure_destination_t
1678       \str_case:nnF {#2}
1679       {
1680         { xyz } { /XYZ ~ @xpos ~ @ypos ~ null }
1681         { fit } { /Fit }
1682         { fitb } { /FitB }
1683         { fitbh } { /FitBH }
1684         { fitbv } { /FitBV ~ @xpos }
1685         { fith } { /FitH ~ @ypos }
1686         { fitv } { /FitV ~ @xpos }
1687         { fitr } { /Fit }
1688       }
1689       { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
1690     ]
1691   }
1692 }

```

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

```

1693 \cs_new_protected:Npn \__pdf_backend_indexed_structure_destination_aux:nnnn #1#2#3#4
1694 {
1695   \vbox_to_zero:n
1696   {
1697     \__kernel_kern:n {#4}
1698     \hbox:n
1699     {
1700       \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
1701       \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
1702     }
1703     \tex_vss:D
1704   }
1705   \__kernel_kern:n {#1}
1706   \vbox_to_zero:n
1707   {
1708     \__kernel_kern:n { -#3 }
1709     \hbox:n
1710     {
1711       \__pdf_backend:n
1712       {
1713         dest ~ (#2)
1714         [
1715           @thispage
1716           /FitR ~
1717           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1718           @xpos ~ @ypos
1719         ]
1720       }

```

Here we add the structure destination to the same box

```

1721       \__pdf_backend:e
1722       {
1723         obj ~ @pdf.SDest.\exp_not:n{#2}
1724         [
1725           \exp_after:wN \pdf_object_ref_indexed:nn \l_pdf_current_structure_destin
1726           /FitR ~
1727           @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
1728           @xpos ~ @ypos
1729         ]
1730       }
1731     }
1732     \tex_vss:D
1733   }
1734   \__kernel_kern:n { -#1 }
1735 }

```

And now we redefine the destination command:

```

1736 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1737 {
1738   \exp_args:Ne \__pdf_backend_indexed_structure_destination_aux:nnnn
1739   { \dim_eval:n {#2} } {#1} {#3} {#4}
1740 }
1741 </xdvipdfmx | dvipdfmx>

```

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

```

1742 <*pdftex>
1743 \bool_lazy_and:nnT
1744 { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
1745 { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
1746 {
1747   \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nn #1#2
1748   {
1749     \tex_pdfdest:D
1750     name {#1}
1751     \str_case:nnF {#2}
1752     {
1753       { xyz } { xyz }
1754       { fit } { fit }
1755       { fitb } { fitb }
1756       { fitbh } { fitbh }
1757       { fitbv } { fitbv }
1758       { fith } { fith }
1759       { fitv } { fitv }
1760       { fitr } { fitr }
1761     }
1762     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1763     \scan_stop:
1764     \tex_pdfdest:D
1765     struct~
1766     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_des
1767     name {#1}
1768     \str_case:nnF {#2}
1769     {
1770       { xyz } { xyz }
1771       { fit } { fit }
1772       { fitb } { fitb }
1773       { fitbh } { fitbh }
1774       { fitbv } { fitbv }
1775       { fith } { fith }
1776       { fitv } { fitv }
1777       { fitr } { fitr }
1778     }
1779     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1780     \scan_stop:
1781   }
1782 \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1783 {
1784   \tex_pdfdest:D
1785   name {#1}
1786   fitr ~
1787   width \dim_eval:n {#2} ~
1788   height \dim_eval:n {#3} ~
1789   depth \dim_eval:n {#4} \scan_stop:
1790   \tex_pdfdest:D
1791   struct~
1792   \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_destinati
1793   name {#1}
1794   fitr ~

```

```

1795     width \dim_eval:n {#2} ~
1796     height \dim_eval:n {#3} ~
1797     depth \dim_eval:n {#4} \scan_stop:
1798   }
1799 }
1800 </pdfTeX>

```

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

```

1801 <*luatex>
1802 \int_compare:nNtT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
1803 {
1804   \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nn #1#2
1805   {
1806     \tex_pdfextension:D dest
1807     name {#1}
1808     \str_case:nnF {#2}
1809     {
1810       { xyz } { xyz }
1811       { fit } { fit }
1812       { fitb } { fitb }
1813       { fitbh } { fitbh }
1814       { fitbv } { fitbv }
1815       { fith } { fith }
1816       { fitv } { fitv }
1817       { fitr } { fitr }
1818     }
1819     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1820     \scan_stop:
1821     \tex_pdfextension:D dest
1822     struct~
1823     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_desti
1824     name {#1}
1825     \str_case:nnF {#2}
1826     {
1827       { xyz } { xyz }
1828       { fit } { fit }
1829       { fitb } { fitb }
1830       { fitbh } { fitbh }
1831       { fitbv } { fitbv }
1832       { fith } { fith }
1833       { fitv } { fitv }
1834       { fitr } { fitr }
1835     }
1836     { xyz ~ zoom \fp_eval:n { #2 * 10 } }
1837     \scan_stop:
1838   }
1839   \cs_set_protected:Npn \__pdf_backend_indexed_structure_destination:nnnn #1#2#3#4
1840   {
1841     \tex_pdfextension:D dest
1842     name {#1}
1843     fitr ~
1844     width \dim_eval:n {#2} ~
1845     height \dim_eval:n {#3} ~
1846     depth \dim_eval:n {#4} \scan_stop:
1847     \tex_pdfextension:D dest

```



```

1848     struct~
1849     \exp_after:wN \__kernel_pdf_object_id_indexed:nn \l_pdf_current_structure_destinati
1850     name {#1}
1851     fitr ~
1852     width \dim_eval:n {#2} ~
1853     height \dim_eval:n {#3} ~
1854     depth \dim_eval:n {#4} \scan_stop:
1855   }
1856   \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
1857   {
1858     \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
1859   }
1860 }
1861 </luatex>

```

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

1.12 Settings for regression tests

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

```

1862 <*drivers>
1863 \cs_new_protected:Npn \__pdf_backend_set_regression_data:
1864 {
1865   \sys_gset_rand_seed:n{1000}
1866   \pdfmanagement_add:nnn{Info}{Creator}{(TeX)}
1867 </drivers>
1868 <*dvips>
1869   \AddToHook{begindocument}{\pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX+dvips)}}
1870   \__kernel_backend_literal:e{!~<</DocumentUUID~(DocumentUUID)>>~setpagedevice}
1871   \__kernel_backend_literal:e{!~<</InstanceUUID~(InstanceUUID)>>~setpagedevice}
1872   \pdfmanagement_add:nne{Info}{CreationDate}{(\c_sys_timestamp_str)}
1873   \pdfmanagement_add:nne{Info}{ModDate}{(\c_sys_timestamp_str)}
1874 </dvips>
1875 <*dviPDFMx>
1876   \pdfmanagement_add:nnn{Info}{Producer}{(dviPDFMx)}
1877   \__kernel_backend_literal:e
1878     {pdf:trailerid [~
1879       <00112233445566778899aabbccddeeff>~
1880       <00112233445566778899aabbccddeeff>~
1881     ]}
1882 </dviPDFMx>
1883 <*xdviPDFMx>
1884   \pdfmanagement_add:nnn{Info}{Producer}{(xetex)}
1885   \__kernel_backend_literal:e
1886     {pdf:trailerid [~
1887       <00112233445566778899aabbccddeeff>~
1888       <00112233445566778899aabbccddeeff>~
1889     ]}
1890 </xdviPDFMx>
1891 <*pdftex>
1892   \pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX)}

```

```

1893 \tex_pdfsuppressptexinfo:D 7 \scan_stop:
1894 \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
1895 </pdftex>
1896 <*luatex>
1897 \pdfmanagement_add:nnn{Info}{Producer}{(LuaTeX)}
1898 \tex_pdfvariable:D suppressoptionalinfo 7\relax
1899 \tex_pdfvariable:D trailerid
1900   {[~
1901     <2350CAD05F8A7AF0AA4058486855344F>~
1902     <2350CAD05F8A7AF0AA4058486855344F>~
1903   ]}
1904 </luatex>

```

Embedded files should also have a fix date.

```

1905 <*drivers>
1906 \pdfdict_put:nne {l_pdffile/Params} {ModDate}{(\c_sys_timestamp_str)}
1907 \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-8c19-6237d832f}
1908 \AddToDocumentProperties[hyperref]{pdfproducer}{\c_sys_engine_exec_str-NN.NN.NN}
1909   }
1910 </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`:

```

1911 <*luatex>
1912 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1913   {
1914     \tex_immediate:D \tex_pdfextension:D obj ~uncompressed~
1915     \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1916   }
1917 </luatex>
1918 <*pdftex>
1919 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1920   {
1921     \group_begin:
1922     \tex_pdfcompresslevel:D 0 \scan_stop:
1923     \tex_immediate:D \tex_pdfobj:D
1924     \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
1925     \group_end:
1926   }
1927 </pdftex>
1928 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1929 \cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
1930   {
1931     \pdf_object_unnamed_write:nn {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
1932   }
1933 </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.

```
1934 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1935 \cs_new_protected:Npn \_pdf_backend_omit_charset:n #1 {} % #1 number
1936 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1937 <*pdftex>
1938 \cs_new_protected:Npn \_pdf_backend_omit_charset:n #1 % #1 number
1939 {
1940   \tex_pdfomitcharset:D = #1 \scan_stop:
1941 }
1942 </pdftex>
1943 <*luatex>
1944 \cs_new_protected:Npn \_pdf_backend_omit_charset:n #1 % #1 number
1945 {
1946   \tex_pdfvariable:D omitcharset = #1 \scan_stop:
1947 }
1948 </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.

```
1949 <*xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1950 \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 {} % #1 number
1951 </xdvipdfmx | dvipdfmx | dvips | dvisvgm>
1952 <*pdftex>
1953 \bool_lazy_and:nnTF
1954 { \int_compare_p:nNn {\tex_pdfversion:D } > {139} }
1955 { \int_compare_p:nNn {\tex_pdfversion:D } > {24} }
1956 {
1957   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 % #1 number
1958   {
1959     \pdfomitinfodict = #1 \scan_stop:
1960   }
1961 }
1962 {
1963   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 {} % #1 number
1964 }
1965 }
1966 </pdftex>
1967 <*luatex>
1968 \int_compare:nNnTF {\directlua{tex.print(status.list()["development_id"])} } > {7560}
1969 {
1970   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 % #1 number
1971   {
1972     \tex_pdfvariable:D omitinfodict = #1 \scan_stop:
1973   }
1974 }
1975 {
1976   \cs_new_protected:Npn \_pdf_backend_omit_info:n #1 {} % #1 number
```

```

1977 }
1978 </luatex>

```

(End of definition for __pdf_backend_omit_info:n.)

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

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

```

1979 <*\xdvipdfmx | dvipdfmx | dvips | dvisvgm | pdftex>
1980 \cs_new_protected:Npn \__pdf_backend_omit_cidset:n #1 {} % #1 number
1981 </\xdvipdfmx | dvipdfmx | dvips | dvisvgm | pdftex>
1982 <*\luatex>
1983 \cs_new_protected:Npn \__pdf_backend_omit_cidset:n #1 % #1 number
1984 {
1985     \tex_pdfvariable:D omitcidset = #1 \scan_stop:
1986 }
1987 </luatex>

```

(End of definition for __pdf_backend_omit_cidset:n.)

1.15 lua code for lualatex

```

1988 <*\lua>
1989 ltx= ltx or {}
1990 ltx.__pdf = ltx.__pdf or {}
1991 ltx.__pdf.Page = ltx.__pdf.Page or {}
1992 ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
1993 ltx.__pdf.Page.Resources = ltx.__pdf.Page.Resources or {}
1994 ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
1995 ltx.__pdf.Page.Resources.List={"ExtGState", "ColorSpace", "Pattern", "Shading"}
1996 ltx.__pdf.object = ltx.__pdf.object or {}
1997
1998 ltx.pdf= ltx.pdf or {} -- for "public" functions
1999
2000 local __pdf = ltx.__pdf
2001 local pdf = pdf
2002
2003 local function __pdf_backend_Page_gput (name,value)
2004     __pdf.Page.dflt[name]=value
2005 end
2006
2007 local function __pdf_backend_Page_gremove (name)
2008     __pdf.Page.dflt[name]=nil
2009 end
2010
2011 local function __pdf_backend_Page_gclear ()
2012     __pdf.Page.dflt={}
2013 end
2014
2015 local function __pdf_backend_ThisPage_gput (page,name,value)
2016     __pdf.Page[page] = __pdf.Page[page] or {}
2017     __pdf.Page[page][name]=value
2018 end
2019
2020 local function __pdf_backend_ThisPage_gpush (page)

```

```

2021 local token=""
2022 local t = {}
2023 local tkeys= {}
2024 for name,value in pairs(__pdf.Page.dflt) do
2025     t[name]=value
2026 end
2027 if __pdf.Page[page] then
2028     for name,value in pairs(__pdf.Page[page]) do
2029         t[name] = value
2030     end
2031 end
2032 -- sort the table to get reliable test files.
2033 for name,value in pairs(t) do
2034     table.insert(tkeys,name)
2035 end
2036 table.sort(tkeys)
2037 for _,name in ipairs(tkeys) do
2038     token = token .. "/"..name.." "..t[name]
2039 end
2040 return token
2041 end
2042
2043 function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly"]
2044     __pdf_backend_ThisPage_gput (page,name,value)
2045 end
2046
2047 function ltx.__pdf.backend_ThisPage_gpush (page)
2048     pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
2049 end
2050
2051 function ltx.__pdf.backend_Page_gput (name,value)
2052     __pdf_backend_Page_gput (name,value)
2053 end
2054
2055 function ltx.__pdf.backend_Page_gremove (name)
2056     __pdf_backend_Page_gremove (name)
2057 end
2058
2059 function ltx.__pdf.backend_Page_gclear ()
2060     __pdf_backend_Page_gclear ()
2061 end
2062
2063
2064 local Properties = ltx.__pdf.Page.Resources.Properties
2065 local ResourceList= ltx.__pdf.Page.Resources.List
2066 local function __pdf_backend_PageResources_gpush (page)
2067     local token=""
2068     if Properties[page] then
2069         -- we sort the table, so that the pdf test works
2070         local t = {}
2071         for name,value in pairs (Properties[page]) do
2072             table.insert (t,name)
2073         end
2074         table.sort (t)

```

```

2075 for _,name in ipairs(t) do
2076   token = token .. "/"..name.." ".. Properties[page][name]
2077   end
2078   token = "/Properties <<"..token..">>"
2079 end
2080 for i,name in ipairs(ResourceList) do
2081   if ltx.__pdf.Page.Resources[name] then
2082     token = token .. "/"..name.." "..ltx.pdf.object_ref("__pdf/Page/Resources/"..name)
2083   end
2084 end
2085 return token
2086 end
2087
2088 -- the function is public, as I probably need it in tagpdf too ...
2089 function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
2090 Properties[page] = Properties[page] or {}
2091 Properties[page][name]=value
2092 pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
2093 end
2094
2095 function ltx.pdf.Page_Resources_gpush(page)
2096 pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
2097 end
2098
2099 function ltx.pdf.object_ref (objname)
2100 if ltx.__pdf.object[objname] then
2101   local ref= ltx.__pdf.object[objname]
2102   return ref
2103 else
2104   return "false"
2105 end
2106 end
2107 </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> 1907, 1908	<code>\box_dp:N</code> 1005, 1086, 1178, 1281, 1295
<code>\AddToHook</code> 1869	<code>\box_ht:N</code> 1002, 1083, 1175, 1278, 1290
	<code>\box_new:N</code> 88, 89, 1164
B	<code>\box_scale:Nnn</code> 1299
bool commands:	<code>\box_set_dp:Nn</code> 1179, 1245, 1342, 1366
<code>\bool_if:NTF</code> 39, 533, 569,	<code>\box_set_ht:Nn</code> 1180, 1244, 1343, 1365
650, 656, 692, 716, 751, 775, 805, 844	<code>\box_set_wd:Nn</code> 1181, 1243, 1344, 1364
<code>\bool_lazy_and:nnTF</code> 1512, 1743, 1953	<code>\box_use:N</code> 1348
<code>\bool_new:N</code> 522	<code>\box_use_drop:N</code> 1192, 1246, 1323, 1367
<code>\bool_set_true:N</code> 992, 1074, 1167, 1268	<code>\box_wd:N</code> 999, 1080, 1172, 1275, 1285

C

clist commands:
 \clist_const:Nn 420
 \clist_map_function:NN 882
 \clist_map_inline:Nn 429, 463, 479, 672
cs commands:
 \cs_generate_variant:Nn
 28, 31, 32, 35, 36, 79, 80, 417
 \cs_gset_eq:NN 651,
 652, 1392, 1393, 1394, 1398, 1399, 1400
 \cs_if_exist:NTF 432, 948
 \cs_if_free:NTF 941
 \cs_new:Npn 74, 100, 106,
 248, 858, 1058, 1140, 1229, 1253, 1369
 \cs_new_protected:Npn ... 41, 45,
 55, 68, 150, 159, 175, 181, 187, 194,
 201, 210, 230, 253, 263, 277, 289,
 306, 317, 324, 331, 340, 349, 356,
 363, 370, 379, 388, 396, 399, 405,
 410, 413, 444, 455, 461, 487, 491,
 503, 506, 507, 511, 514, 515, 519,
 535, 558, 582, 670, 765, 867, 891,
 898, 905, 914, 918, 921, 924, 936,
 943, 944, 984, 1051, 1066, 1131,
 1154, 1234, 1251, 1252, 1259, 1353,
 1390, 1396, 1455, 1693, 1863, 1912,
 1919, 1929, 1935, 1938, 1944, 1950,
 1957, 1963, 1970, 1976, 1980, 1983
 \cs_new_protected:Npx 169
 \cs_set_eq:NN
 . 530, 531, 566, 567, 662, 749, 757,
 842, 850, 1404, 1405, 1406, 1407, 1408
 \cs_set_protected:Npn .. 526, 542,
 546, 550, 554, 564, 571, 574, 576,
 578, 580, 593, 612, 631, 637, 643,
 648, 658, 664, 687, 711, 735, 739,
 744, 753, 760, 768, 798, 829, 833,
 838, 846, 853, 950, 954, 963, 967,
 974, 978, 1411, 1501, 1506, 1516,
 1555, 1576, 1585, 1624, 1645, 1652,
 1736, 1747, 1782, 1804, 1839, 1856

D

dim commands:
 \dim_eval:n 1504, 1560,
 1561, 1562, 1571, 1572, 1573, 1629,
 1630, 1631, 1640, 1641, 1642, 1739,
 1787, 1788, 1789, 1795, 1796, 1797,
 1844, 1845, 1846, 1852, 1853, 1854
 \dim_to_decimal_in_sp:n
 1285, 1290, 1295
 \c_zero_dim
 .. 1179, 1180, 1181, 1342, 1343, 1344
\directlua 97, 1583, 1802, 1968

E

exp commands:
 \exp_after:wN
 .. 1677, 1725, 1766, 1792, 1823, 1849
 \exp_args:Ne . 700, 724, 1432, 1438,
 1483, 1489, 1503, 1533, 1538, 1563,
 1568, 1602, 1607, 1632, 1637, 1738
 \exp_args:NNe 869
 \exp_not:n 617, 715,
 802, 1415, 1436, 1487, 1656, 1675, 1723

F

fp commands:
 \fp_eval:n
 1429, 1450, 1531, 1551, 1600, 1620,
 1670, 1689, 1762, 1779, 1819, 1836

G

group commands:
 \group_begin: 1921
 \group_end: 1925

H

hbox commands:
 \hbox:n 1460, 1471, 1698, 1709
 \hbox_gset:Nn 1165
 \hbox_set:Nn
 ... 990, 1072, 1236, 1266, 1300, 1355
hook commands:
 \hook_gput_code:nnn .. 142, 482, 1345
 \hook_gput_next_code:nn 1182
 \hook_gset_rule:nmm 476, 477

I

int commands:
 \int_compare:nNnTF
 ... 961, 1013, 1094, 1583, 1802, 1968
 \int_compare_p:nNn
 .. 1513, 1514, 1744, 1745, 1954, 1955
 \int_const:Nn . 1046, 1126, 1161, 1262
 \int_gincr:N 213, 595, 614,
 689, 713, 770, 774, 800, 804, 1160, 1261
 \int_if_exist:NTF 1379
 \int_new:N 92, 93, 94
 \int_use:N 214, 217,
 598, 606, 617, 625, 691, 696, 705,
 715, 720, 729, 772, 779, 783, 786,
 794, 802, 809, 813, 816, 824, 1054,
 1060, 1133, 1141, 1231, 1309, 1336,
 1360, 1371, 1537, 1567, 1606, 1636

K

kernel internal commands:
 _kernel_backend_literal:n
 .. 31, 83, 596, 600, 615, 619, 633,
 645, 666, 676, 1870, 1871, 1877, 1885

__pdf_backend_link_off:	__pdf_backend_record_abspage:n
. 941, 943, 950, 963, 974 68, 79, 214, 783, 813
__pdf_backend_link_on:	__pdf_backend_ref_abspage:n
. 944, 954, 967, 978 74, 80, 217, 786, 816
__pdf_backend_luastring:n	\g__pdf_backend_resourceid_int
163, 248, 257, 269, 270, 281, 296, 297 91, 213, 214, 217, 774, 779,
__pdf_backend_metadata_stream:n	783, 786, 794, 804, 809, 813, 816, 824
. 1912, 1919, 1929	__pdf_backend_set_regression_-
\g__pdf_backend_name_int	data: 1863
. 91, 595, 598, 606,	__pdf_backend_shipout_bdc:nn
614, 617, 625, 689, 691, 696, 705, 13, 521, 571
713, 715, 720, 729, 770, 772, 800, 802	__pdf_backend_structure_-
__pdf_backend_Names_gpush:nn	destination:nn
. 891, 898, 905, 914, 918 1392, 1404, 1410, 1411, 1516, 1585
__pdf_backend_NamesEmbeddedFiles_-	__pdf_backend_structure_-
add:nn 920, 921, 924, 936	destination:nnnn
\g__pdf_backend_object_int 1393, 1405, 1410, 1501, 1555, 1624
. 1160, 1163, 1261, 1264, 1309	__pdf_backend_structure_-
__pdf_backend_object_last:	destination_aux:nnnn 1455, 1503
. 548, 626, 721, 730, 810, 825	__pdf_backend_ThisPage_gpush:n
__pdf_backend_object_write:nn 6, 184, 230, 306, 349, 388, 413
. 1915, 1924	__pdf_backend_ThisPage_gput:nn
__pdf_backend_omit_charset:n 6, 184, 210, 289, 340, 379, 410
. 1934, 1935, 1938, 1944	\g__pdf_backend_thispage_-
__pdf_backend_omit_cidset:n	shipout_tl 6
. 1979, 1980, 1983	\l__pdf_backend_tmpa_box
__pdf_backend_omit_info:n 85, 990, 999, 1002, 1005, 1045,
. 1949, 1950, 1957, 1963, 1970, 1976	1072, 1080, 1083, 1086, 1125, 1236,
__pdf_backend_Page_gput:nn	1243, 1244, 1245, 1246, 1266, 1275,
. 6, 184, 194, 263, 324, 363, 399	1278, 1281, 1285, 1290, 1295, 1299,
__pdf_backend_Page_gremove:n	1323, 1355, 1364, 1365, 1366, 1367
. 6, 184, 201, 277, 331, 370, 405	\l__pdf_backend_tmpb_box
\g__pdf_backend_page_int 91 89, 1300, 1342, 1343, 1344, 1348
__pdf_backend_Page_primitive:n	\l__pdf_backend_xform_bool
. 6, 184, 187, 240, 253, 522, 692,
317, 342, 351, 356, 381, 390, 396, 417	716, 775, 805, 992, 1074, 1167, 1268
__pdf_backend_PageResources:n	__pdf_backend_xform_if_exist:n
. 487, 506, 514 1377, 1383
\c__pdf_backend_PageResources_-	__pdf_backend_xform_new:nnnn
clist 419, 429, 463, 479, 672, 883 983, 984, 1066, 1154, 1251, 1259
__pdf_backend_PageResources_-	__pdf_backend_xform_ref:n
gpush:n 983, 1058,
. 13, 521, 558, 582, 670, 765, 867	1140, 1187, 1229, 1240, 1253, 1369
__pdf_backend_PageResources_-	\l__pdf_backend_xform_tmpdp_tl
gpush_aux:n 858, 884 1257, 1293, 1307, 1314
__pdf_backend_PageResources_-	\l__pdf_backend_xform_tmpht_tl
gput:nnn 428, 444, 455, 491, 507, 515 1258, 1288, 1312
__pdf_backend_PageResources_-	\l__pdf_backend_xform_tmpwd_tl
obj_gpush: 428, 461, 503, 511, 519 1256, 1283, 1313
__pdf_backend_Pages_primitive:n	__pdf_backend_xform_use:n
. 149, 150, 159, 169, 175, 181 983, 1051, 1131, 1234, 1252, 1353
__pdf_backend_pdfmark:n	\g__pdf_tmpa_prop 85, 232, 237, 242
. 36, 528, 544, 548, 552, 556, 926	\l__pdf_tmpa_tl
 85, 215, 219, 221, 224, 784,

	788, 790, 793, 814, 818, 820, 823, 826		
pdfdict commands:			
\pdfdict_gput:nnn	196, 224, 326, 365, 401, 446, 457, 509, 517, 694, 718, 777, 792, 807, 822		
\pdfdict_gremove:nn	203, 333, 372, 407		
\pdfdict_if_exist:nTF	219, 788, 818		
\pdfdict_item:nn	242, 863, 878		
\pdfdict_new:n	221, 790, 820		
\pdfdict_put:nnm	1906		
\pdfdict_show:n	826		
\pdfdict_use:n	352, 391, 470, 1020, 1101		
\pdfextension	965, 969		
\pdfliteral	2		
pdfmanagement commands:			
\pdfmanagement_add:nnn	1866, 1869, 1872, 1873, 1876, 1884, 1892, 1897		
pdfmanagement internal commands:			
\g_pdfmanagement_active_bool	650		
\l_pdfmanagement_delayed_			
shipout_bool	39, 533, 569, 656, 751, 844		
\pdfnames	20		
\pdfomitinfodict	1959		
\pdfpageref	3		
\pdfrunninglinkoff	948, 952		
\pdfrunninglinkon	956		
\pdftrailerid	1894		
pdfxform commands:			
\pdfxform_dp:n	1190, 1245, 1366		
\pdfxform_ht:n	1189, 1244, 1365		
\pdfxform_if_exist:n	1383		
\pdfxform_wd:n	1188, 1243, 1364		
prg commands:			
\prg_new_conditional:Npnn	1377		
\prg_new_eq_conditional:NNn	1383		
\prg_return_false:	1381		
\prg_return_true:	1380		
prop commands:			
\prop_count:N	1014, 1095		
\prop_gclear:N	993, 1075, 1269		
\prop_gput:Nnn	237, 494		
\prop_gset_eq:NN	232		
\prop_if_empty:NTF	465, 674, 860, 1024, 1029, 1034, 1039, 1104, 1109, 1114, 1119		
\prop_if_exist:NTF	233, 871		
\prop_map_function:NN	242, 876		
\prop_map_inline:Nn	235		
\prop_new:N	86		
property commands:			
\property_record:nn	71		
\property_ref:nn	76		
\ProvidesExplFile	1		
		R	
		\relax	135, 1898
		S	
		scan commands:	
		\scan_stop:	1055, 1137, 1532, 1552, 1562, 1573, 1601, 1621, 1631, 1642, 1763, 1780, 1789, 1797, 1820, 1837, 1846, 1854, 1893, 1922, 1940, 1946, 1959, 1972, 1985
		\special	2
		str commands:	
		\str_case:nnTF	1418, 1439, 1520, 1540, 1589, 1609, 1659, 1678, 1751, 1768, 1808, 1825
		\str_convert_pdfname:n	102, 495
		\str_if_eq:nnTF	1317, 1328
		sys commands:	
		\c_sys_engine_exec_str	1908
		\sys_gset_rand_seed:n	1865
		\sys_if_engine luatex:TF	157
		\c_sys_timestamp_str	1872, 1873, 1906
		T	
		TeX and L ^A T _E X 2 _ε commands:	
		\@bsphack	70
		\@esphack	72
		\@kernel@after@enddocument@afterlastpage	112, 113
		\@kernel@after@shipout@background	133, 136
		\@kernel@after@shipout@lastpage	119, 120, 126, 127
		\@kernel@before@shipout@background	135
		\g@addto@macro	135, 136
		\special	2
		tex commands:	
		\tex_directlua:D	161, 265, 279, 432, 434, 447, 448
		\tex_global:D	152, 189, 869
		\tex_immediate:D	1007, 1088, 1914, 1923
		\tex_latelua:D	255, 291, 308, 700, 724
		\tex_luaescapestring:D	250
		\tex luatexversion:D	961
		\tex_pdfcompresslevel:D	1922
		\tex_pdfdest:D	1518, 1535, 1557, 1565, 1749, 1764, 1784, 1790
		\tex_pdfextension:D	48, 58, 901, 1587, 1604, 1626, 1634, 1806, 1821, 1841, 1847, 1914
		\tex_pdflastxform:D	1048, 1128
		\tex_pdfliteral:D	51, 61
		\tex_pdfnames:D	894

