

The `postnotes` package

Code documentation

`gusbrs`

<https://github.com/gusbrs/postnotes>
<https://www.ctan.org/pkg/postnotes>

Version v0.4.0 – 2024-11-04

Contents

1	Initial setup	2
2	Data	2
3	Options	7
4	<code>\postnote</code>	15
5	<code>\postnoteref</code>	23
6	<code>\postnotesection</code>	24
7	<code>\printpostnotes</code>	25
8	Headers	36
9	Compatibility	42
10	Languages	61
	Index	64

1 Initial setup

Start the DocStrip guards.

```
1 <*package>
   Identify the internal prefix (LATEX3 DocStrip convention).
2 <@@=postnotes>
```

The new syntax for file/package hooks, which the package assumes, requires kernel 2021-11-15 (`ltnews34`, `lfilehook`). Furthermore, the kernel of 2022-06-01 introduced a couple of very nice features which simplifies the relation with `hyperref` (`ltnews35`, `hyperref-linktarget`): the provision of `\MakeLinkTarget` and the definition by the kernel of the starred version of `\ref`, which we can use regardless of `hyperref` being loaded. Also, since we followed the move to e-type expansion, to play safe we require the 2023-11-01 kernel or newer. Finally, the tagging sockets and block code required for tagging support need the 2024-06-01 kernel.

```
3 \def\postnotes@required@kernel{2024-06-01}
4 \NeedsTeXFormat{LaTeX2e}[\postnotes@required@kernel]
5 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
6 \IfFormatAtLeastTF{\postnotes@required@kernel}
7   {}
8   {%
9     \PackageError{postnotes}{LaTeX kernel too old}
10    {%
11      'postnotes' requires a LaTeX kernel \postnotes@required@kernel\space or newer.%
12    }%
13  }%
14 \ProvidesExplPackage {postnotes} {2024-11-04} {0.4.0}
15 {Endnotes for LaTeX}
```

`\l__postnotes_tmpa_tl` Temporary scratch variables.

```
\l__postnotes_tmpb_tl
\l__postnotes_tmpa_seq
\l__postnotes_tmpb_seq
\l__postnotes_tmpa_box
```

```
16 \tl_new:N \l__postnotes_tmpa_tl
17 \tl_new:N \l__postnotes_tmpb_tl
18 \seq_new:N \l__postnotes_tmpa_seq
19 \seq_new:N \l__postnotes_tmpb_seq
20 \box_new:N \l__postnotes_tmpa_box
```

(End of definition for `\l__postnotes_tmpa_tl` and others.)

2 Data

`__postnotes_data_name:n` Returns the name of the property list variable which stores the data of the `\postnote` with `<note id>` number.

```
\__postnotes_data_name:n {<note id>}
21 \cs_new:Npn \__postnotes_data_name:n #1
22   { g__postnotes_#1_data_prop }
23 \cs_generate_variant:Nn \__postnotes_data_name:n { e }
```

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

`postnotes` provides a number of hooks from the new hook system to grant some points of access in key places of the package. Note that hooks created with `\NewHook` are meant to be public interfaces (see <https://chat.stackexchange.com/transcript/message/62955941#62955941>, and following discussion).

`__postnotes_store:nn` Stores the metadata and \langle *note content* \rangle of `\postnote` with ID \langle *note id* \rangle , from where it is called. The `postnotes/note/store` hook is intended to add further data to the note, when required to support packages with specific needs.

```
\__postnotes_store:nn { $\langle$ note id $\rangle$ } { $\langle$ note content $\rangle$ }  
  
24 \NewHook { postnotes/note/store }  
25 \cs_new_protected:Npn \__postnotes_store:nn #1#2  
26 {  
27   \prop_new:c { \__postnotes_data_name:e {#1} }  
28   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { note }  
29   \prop_gput:cne { \__postnotes_data_name:e {#1} } { mark }  
30     { \l__postnotes_mark_tl }  
31   \prop_gput:cne { \__postnotes_data_name:e {#1} } { counter }  
32     { \int_use:N \c@postnote }  
33   \prop_gput:cne { \__postnotes_data_name:e {#1} } { sortnum }  
34     {  
35       \bool_if:NTF \l__postnotes_manual_sortnum_bool  
36         { \fp_use:N \l__postnotes_sort_num_fp }  
37         { \int_use:N \c@postnote }  
38     }  
39   \cs_if_exist:cT { chapter }  
40     {  
41       \prop_gput:cne { \__postnotes_data_name:e {#1} }  
42         { thechapter } { \thechapter }  
43     }  
44   \prop_gput:cne { \__postnotes_data_name:e {#1} } { thesection }  
45     { \thesection }  
46   \prop_gput:cne { \__postnotes_data_name:e {#1} } { pnsectname }  
47     { \g__postnotes_section_name_tl }  
48   \prop_gput:cne { \__postnotes_data_name:e {#1} } { pnsectid }  
49     { \int_use:N \g__postnotes_sectid_int }  
50   \prop_gput:cne { \__postnotes_data_name:e {#1} } { multibool }  
51     { \bool_to_str:N \l__postnotes_maybe_multi_bool }  
52   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}  
53   \UseHook { postnotes/note/store }  
54 }
```

(End of definition for `__postnotes_store:nn`.)

`__postnotes_store_section:nn` Stores the metadata and \langle *note content* \rangle of `\postnotessection` with ID \langle *note id* \rangle , from where it is called.

```
\__postnotes_store_section:nn { $\langle$ note id $\rangle$ } { $\langle$ note content $\rangle$ }  
  
55 \cs_new_protected:Npn \__postnotes_store_section:nn #1#2  
56 {  
57   \prop_new:c { \__postnotes_data_name:e {#1} }  
58 }
```

```

58 \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { section }
59 \cs_if_exist:cT { chapter }
60 {
61   \prop_gput:cne { \__postnotes_data_name:e {#1} }
62     { thechapter } { \thechapter }
63 }
64 \prop_gput:cne { \__postnotes_data_name:e {#1} } { thesection }
65 { \thesection }
66 \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}
67 }
68 \cs_generate_variant:Nn \__postnotes_store_section:nn { ne }

```

(End of definition for __postnotes_store_section:nn.)

__postnotes_prop_get:nnN Convenience functions to retrieve and clear data from a note based on the ID number.

__postnotes_prop_item:nn
__postnotes_prop_gclear:n

```

\__postnotes_prop_get:nnN {<note id>} {<property>} {<tl var to set>}
\__postnotes_prop_item:nn {<note id>} {<property>}
\__postnotes_prop_gclear:n {<note id>}

69 \cs_new_protected:Npn \__postnotes_prop_get:nnN #1#2#3
70 {
71   \prop_get:cnNF { \__postnotes_data_name:e {#1} } {#2} #3
72   { \tl_clear:N #3 }
73 }
74 \cs_new:Npn \__postnotes_prop_item:nn #1#2
75 { \prop_item:cn { \__postnotes_data_name:e {#1} } {#2} }
76 \cs_new_protected:Npn \__postnotes_prop_gclear:n #1
77 { \prop_gclear:c { \__postnotes_data_name:e {#1} } }

```

(End of definition for __postnotes_prop_get:nnN, __postnotes_prop_item:nn, and __postnotes_prop_gclear:n.)

\post@note
_postnotes_store_labelseq:nn
_postnotes_step_counteraux:nnn

\post@note is the \newlabel equivalent for postnotes. Based on the kernel's \@newl@bel so that we get L^AT_EX checks for multiple and changed references for free (procedure learnt from zref). \post@note, when the .aux file is read, defines macros named \postnote@r@<label type>@<id>, according to the prefix set by \c__postnotes_ref_prefix_tl. __postnotes_store_labelseq:nn is an auxiliary function to store the sequence of the labels in the .aux file, used to check for possible sorting problems due to floats. __postnotes_step_counteraux:nnn handles counter stepping and storing for the counteraux option.

```

\post@note {<label type>} {<id>} {<label content (page)>}
  {<counteraux step>}
\__postnotes_store_labelseq:nn {<label type>} {<id>}
\__postnotes_step_counteraux:nnn {<label type>} {<id>}
  {<counteraux step>}

78 \tl_const:Nn \c__postnotes_ref_prefix_tl { postnote@r }
79 \seq_new:N \g__postnotes_labelseq_seq
80 \int_new:N \g__postnotes_postnote_counteraux_int
81 \prop_new:N \g__postnotes_counteraux_prop
82 \bool_new:N \g__postnotes_firstrun_bool
83 \bool_gset_true:N \g__postnotes_firstrun_bool

```

```

84 \cs_new_protected:Npn \__postnotes_store_labelseq:nn #1#2
85 {
86   \bool_lazy_any:nT
87   {
88     { \str_if_eq_p:nn {#1} { mark } }
89     { \str_if_eq_p:nn {#1} { section } }
90     { \str_if_eq_p:nn {#1} { preprint } }
91   }
92   { \seq_gput_right:Nn \g__postnotes_labelseq_seq { {#1} {#2} } }
93 }
94 \cs_new_protected:Npn \__postnotes_step_counter:aux:nnn #1#2#3
95 {
96   \bool_lazy_and:nnT
97   { \g__postnotes_counter:aux_bool }
98   { \str_if_eq_p:nn {#1} { mark } }
99   {
100     \int_gadd:Nn \g__postnotes_postnote_counter:aux_int { #3 }
101     \prop_gput:Nne \g__postnotes_counter:aux_prop { #2 }
102     { \int_use:N \g__postnotes_postnote_counter:aux_int }
103   }
104 }
105 \cs_new_protected:Npe \post@note #1#2#3#4
106 {
107   \exp_not:N \bool_gset_false:N \exp_not:N \g__postnotes_firstrun_bool
108   \exp_not:N \__postnotes_store_labelseq:nn { #1 } { #2 }
109   \exp_not:N \__postnotes_step_counter:aux:nnn { #1 } { #2 } { #4 }
110   \exp_not:N \@newl@bel { \c__postnotes_ref_prefix_tl } { #1 @ #2 } { #3 }
111 }

```

(End of definition for `\post@note`, `__postnotes_store_labelseq:nn`, and `__postnotes_step_counter:aux:nnn`.)

And ensure `\post@note`, `\postnote@setcounter:aux`, and `\postnote@addtocounter:aux` are defined in the `.aux` file. The hooks are the same used by `hyperref` for similar purpose.

```

112 \AddToHook { begindocument }
113 {
114   \legacy_if:nT { @files }
115   {
116     \iow_now:Ne \@mainaux
117     {
118       \token_to_str:N \providecommand
119       \token_to_str:N \post@note [4] { }
120     }
121     \iow_now:Ne \@mainaux
122     {
123       \token_to_str:N \providecommand
124       \token_to_str:N \postnote@setcounter:aux [1] { }
125     }
126     \iow_now:Ne \@mainaux
127     {
128       \token_to_str:N \providecommand
129       \token_to_str:N \postnote@addtocounter:aux [1] { }
130     }
131   }
132 }

```

```

133 \AddToHook { include/before }
134 {
135   \legacy_if:nT { @filesw }
136   {
137     \iow_now:Ne \@mainaux
138     {
139       \token_to_str:N \providecommand
140       \token_to_str:N \post@note [4] { }
141     }
142     \iow_now:Ne \@mainaux
143     {
144       \token_to_str:N \providecommand
145       \token_to_str:N \postnote@setcounteraux [1] { }
146     }
147     \iow_now:Ne \@mainaux
148     {
149       \token_to_str:N \providecommand
150       \token_to_str:N \postnote@addtocounteraux [1] { }
151     }
152   }
153 }

```

`__postnotes_set_label:nnnn` Label setting functions for each pertinent context. They must use `\iow_shipout_e:Nn`, since the main information we are interested in is the page.

```

__postnotes_set_mark_page_label:nn
__postnotes_set_section_page_label:n
__postnotes_set_text_page_label:n
__postnotes_set_print_page_label:n
__postnotes_set_pre_print_label:n

    __postnotes_set_label:nnnn {<label type>} {<note id>} {<value>}
    {<counteraux step>}
    __postnotes_set_mark_page_label:nn {<note id>} {<counteraux step>}
    __postnotes_set_section_page_label:n {<note id>}
    __postnotes_set_text_page_label:n {<note id>}
    __postnotes_set_print_page_label:n {<note id>}
    __postnotes_set_pre_print_label:n {<note id>}

154 \cs_new_protected:Npn __postnotes_set_label:nnnn #1#2#3#4
155 {
156   \legacy_if:nT { @filesw }
157   {
158     \iow_shipout_e:Nn \@auxout
159     { \token_to_str:N \post@note { #1 } { #2 } { #3 } { #4 } }
160   }
161 }
162 \cs_new_protected:Npn __postnotes_set_mark_page_label:nn #1#2
163 { \__postnotes_set_label:nnnn { mark } { #1 } { \thepage } { #2 } }
164 \cs_generate_variant:Nn __postnotes_set_mark_page_label:nn { ee }
165 \cs_new_protected:Npn __postnotes_set_section_page_label:n #1
166 { \__postnotes_set_label:nnnn { section } { #1 } { \thepage } { } }
167 \cs_generate_variant:Nn __postnotes_set_section_page_label:n { e }
168 \cs_new_protected:Npn __postnotes_set_text_page_label:n #1
169 { \__postnotes_set_label:nnnn { text } { #1 } { \int_use:N \c@page } { } }
170 \cs_generate_variant:Nn __postnotes_set_text_page_label:n { e }
171 \cs_new_protected:Npn __postnotes_set_print_page_label:n #1
172 { \__postnotes_set_label:nnnn { print } { #1 } { \int_use:N \c@page } { } }
173 \cs_generate_variant:Nn __postnotes_set_print_page_label:n { e }
174 \cs_new_protected:Npn __postnotes_set_pre_print_label:n #1
175 { \__postnotes_set_label:nnnn { preprint } { #1 } { } { } }

```

```
176 \cs_generate_variant:Nn \__postnotes_set_pre_print_label:n { e }
```

(End of definition for __postnotes_set_label:nnnn and others.)

__postnotes_get_pageref:Nn Reference data extraction functions.

__postnotes_extract_pageref:n

```
\__postnotes_get_pageref:Nn {<tl var to set>} {<label name>}
\__postnotes_extract_pageref:n {<label name>}
```

```
177 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
```

```
178 {
```

```
179   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
```

```
180     { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
```

```
181     { \tl_clear:N #1 }
```

```
182 }
```

```
183 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Ne }
```

```
184 \cs_new:Npn \__postnotes_extract_pageref:n #1
```

```
185 {
```

```
186   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
```

```
187     { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
```

```
188     { \c_empty_tl }
```

```
189 }
```

```
190 \cs_generate_variant:Nn \__postnotes_extract_pageref:n { e }
```

(End of definition for __postnotes_get_pageref:Nn and __postnotes_extract_pageref:n.)

3 Options

heading option

```
191 \keys_define:nn { postnotes/setup }
```

```
192 {
```

```
193   heading .cs_set_protected:Np = \pnheading ,
```

```
194   heading .value_required:n = true ,
```

```
195 }
```

\pnheading Provide default value for \pnheading.

```
196 \cs_if_exist:cTF { chapter }
```

```
197 {
```

```
198   \cs_new_protected:Npn \pnheading
```

```
199     {
```

```
200       \chapter*{\pntitle}
```

```
201       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
```

```
202     }
```

```
203 }
```

```
204 {
```

```
205   \cs_new_protected:Npn \pnheading
```

```
206     {
```

```
207       \section*{\pntitle}
```

```
208       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
```

```
209     }
```

```
210 }
```

(End of definition for \pnheading.)

format option

```
211 \tl_new:N \l__postnotes_print_format_tl
212 \keys_define:nn { postnotes/setup }
213 {
214   format .tl_set:N = \l__postnotes_print_format_tl ,
215   format .initial:n = { \small } ,
216   format .value_required:n = true ,
217 }
```

listenv option

```
218 \tl_new:N \l__postnotes_print_env_tl
219 \bool_new:N \l__postnotes_print_as_list_bool
220 \keys_define:nn { postnotes/setup }
221 {
222   listenv .code:n =
223   {
224     \tl_if_eq:nnTF {#1} { none }
225     {
226       \bool_set_false:N \l__postnotes_print_as_list_bool
227       \tl_set:Nn \l__postnotes_post_printnote_tl { \par }

```

A sensible default just in case. It should not get to be used though.

```
228       \tl_set:Nn \l__postnotes_print_env_tl { itemize }
229     }
230     {
231       \bool_set_true:N \l__postnotes_print_as_list_bool
232       \tl_set:Nn \l__postnotes_print_env_tl {#1}
233     }
234   } ,
235   listenv .initial:n = { postnoteslist } ,
236   listenv .value_required:n = true ,
237 }
```

A couple of built-in list environments provided for convenience, and `postnoteslist` as default. The horizontal setup of the label in these lists is based on the `description` environment of the standard classes (see the *The L^AT_EX Companion*).

```
238 \NewDocumentEnvironment { postnoteslist } { }
239 {
240   \list { }
241   {
242     \setlength { \leftmargin } { Opt }
243     \setlength { \labelwidth } { Opt }
244     \setlength { \itemindent } { .5\parindent }
245     \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n
246     \setlength { \rightmargin } { Opt }
247     \setlength { \listparindent } { \parindent }
248     \setlength { \parsep } { \parskip }
249     \setlength { \itemsep } { Opt }
250     \setlength { \topsep } { .5\topsep }
251     \setlength { \partopsep } { .5\partopsep }
252   }
253 }
254 { \endlist }
255 \NewDocumentEnvironment { postnoteslisthang } { }
```



```

256 {
257   \list { }
258   {
259     \setlength { \leftmargin } { 1em }
260     \setlength { \labelwidth } { -\leftmargin }
261     \setlength { \itemindent } { -2\leftmargin }
262     \cs_set_eq:NN \makelabel \__postnotes_list_makelabel:n
263     \setlength { \rightmargin } { 0pt }
264     \setlength { \listparindent } { \parindent }
265     \setlength { \parsep } { \parskip }
266     \setlength { \itemsep } { 0pt }
267     \setlength { \topsep } { .5\topsep }
268     \setlength { \partopsep } { .5\partopsep }
269   }
270 }
271 { \endlist }
272 \cs_new:Npn \__postnotes_list_makelabel:n #1
273 { \hspace { \labelsep } \normalfont ~ #1 }

```

makemark and maketextmark options

The arguments are: #1 is the mark, #2 and #3 are, respectively, the start and the end of the backlink.

```

274 \keys_define:nn { postnotes/setup }
275 {
276   makemark .cs_set:Np = \__postnotes_make_mark:nnn #1#2#3 ,
277   makemark .value_required:n = true ,

```

From the default kernel definition of \@makefnmark.

```

278   makemark .initial:n =
279     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
280   maketextmark .cs_set:Np = \__postnotes_make_text_mark:nnn #1#2#3 ,
281   maketextmark .value_required:n = true ,
282   maketextmark .initial:n = { #2 #1 . #3 } ,
283 }
284 \cs_generate_variant:Nn \__postnotes_make_mark:nnn { Vnn }

```

pretextmark, posttextmark, postprintnote options

```

285 \tl_new:N \l__postnotes_pre_textmark_tl
286 \tl_new:N \l__postnotes_post_textmark_tl
287 \tl_new:N \l__postnotes_post_printnote_tl
288 \keys_define:nn { postnotes/setup }
289 {
290   pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
291   pretextmark .value_required:n = true ,
292   posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
293   posttextmark .value_required:n = true ,
294   postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
295   postprintnote .value_required:n = true ,
296 }

```

style option

```

297 \keys_define:nn { postnotes/setup }

```

```

298 {
299   style .choice: ,
300   style / endnotes .meta:n =
301   {
302     listenv = none ,
303     format =
304     {
305       \footnotesize
306       \setlength { \rightskip } { Opt }
307       \setlength { \leftskip } { Opt }
308       \setlength { \parindent } { 1.8em }
309     } ,

```

endnotes uses a zero width box to get the desired alignment to the right, but that does not play well with the backlinks, so we have a little more work to do to get this right.

```

310     maketextmark =
311     {
312       \hbox_set:Nn \l__postnotes_tmpa_box
313       { \@textsuperscript { \normalfont ##1 } }
314       \skip_horizontal:n { - \box_wd:N \l__postnotes_tmpa_box }
315       ##2 \box_use:N \l__postnotes_tmpa_box ##3
316     } ,
317   } ,
318   style / pagenote .meta:n =
319   {
320     listenv = none ,
321     format = { } ,
322     pretextmark = { \noindent } ,
323     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
324     posttextmark = { ~ } ,
325   } ,
326 }

```

hyperref and backlink options

```

327 \bool_new:N \l__postnotes_hyperlink_bool
328 \bool_new:N \l__postnotes_hyperref_warn_bool
329 \bool_new:N \l__postnotes_backlink_bool
330 \keys_define:nn { postnotes/setup }
331 {
332   hyperref .choice: ,
333   hyperref / auto .code:n =
334   {
335     \bool_set_true:N \l__postnotes_hyperlink_bool
336     \bool_set_false:N \l__postnotes_hyperref_warn_bool
337   } ,
338   hyperref / true .code:n =
339   {
340     \bool_set_true:N \l__postnotes_hyperlink_bool
341     \bool_set_true:N \l__postnotes_hyperref_warn_bool
342   } ,
343   hyperref / false .code:n =
344   {
345     \bool_set_false:N \l__postnotes_hyperlink_bool
346     \bool_set_false:N \l__postnotes_hyperref_warn_bool

```

```

347     } ,
348     hyperref .initial:n = auto ,
349     hyperref .default:n = true ,
350     backlink .bool_set:N = \l__postnotes_backlink_bool ,
351     backlink .initial:n = true ,
352     backlink .default:n = true ,
353 }
354 \AddToHook { begindocument }
355 {
356   \IfPackageLoadedTF { hyperref }
357   { }
358   {
359     \bool_if:NT \l__postnotes_hyperref_warn_bool
360     { \msg_warning:nn { postnotes } { missing-hyperref } }
361     \bool_set_false:N \l__postnotes_hyperlink_bool
362   }
363   \keys_define:nn { postnotes/setup }
364   {
365     hyperref .code:n =
366     {
367       \msg_warning:nnn { postnotes }
368       { option-preamble-only } { hyperref }
369     } ,
370     backlink .code:n =
371     {
372       \msg_warning:nnn { postnotes }
373       { option-preamble-only } { backlink }
374     } ,
375   }
376 }
377 \msg_new:nnn { postnotes } { option-preamble-only }
378 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
379 \msg_new:nnn { postnotes } { missing-hyperref }
380 { Missing~'hyperref'~package.~Setting~'hyperref=false'. }

```

multiple, multisep options

As far as I can tell, the `multiple` option has its origins in the `footmisc` package, which offers this feature for footnotes. About this option, `footmisc.dtx` observes:

This (revised) code derives from a suggestion by Alexander Rozhenko (the author of the `manyfoot` package): the intention is that `footmisc` and `manyfoot` should be able to ‘interwork’, in the sense that each would recognize the other’s footnote marks and behave appropriately. The trick is that both `\footnote` and `\footnotemark` insert a marker (a cancelling pair of kerns of `\multiplefootnotemark` (of opposite signs), which is detected in following `\footnote` or `\footnotemark` commands.

About the same option, `manyfoot.dtx` notes:

To support `multiple` option from `footmisc` we add the `\FN@mf@prepare` command from `footmisc` (suggested by Frank Mittelbach).

Whatever the exact origin of this feature, the fact is that it has spread throughout the ecosystem, using not only the same basic mechanism but, typically, using *the same variables*: `\multiplefootnotemarker` and `\multifootsep`. With the intention, naturally, that different classes and packages can “interwork” with regard to this feature. And this became a sort of “shared feature” in the ecosystem (the list includes `footmisc`, `KOMA-Script`, `eledmac`, `reledmac`, `tufte`, `memoir`, `parnotes`, `sidenotes`, and now also `postnotes`, see discussion at <https://chat.stackexchange.com/transcript/message/66421777#66421777>). What is crucial for this interplay, however, is not quite the variables themselves, but *the value of the canceling pair of kerns*, stored in `\multiplefootnotemarker`. The fact that the same variables are used by most of the classes and packages which provide the feature speaks for convenience, in that a change in one of them reflects in all participating in the “pool”.

Convenient as it is, this shared use of the same variables can only work as long as the community agrees in what these variables contain to some degree. As far as `\multiplefootnotemarker` goes, I see no divergence, and everybody uses the value of `3sp` for the canceling kerns from `\FN@mf@prepare`. `latex-lab-footnotes.dtx` even documents this value for this purpose in its list of “use of kerns to mark h-mode positions” (see <https://chat.stackexchange.com/transcript/message/66421893#66421893>). Things are not as smooth with regard to `\multifootsep` though. `footmisc` stores a plain comma in `\multifootsep` and applies formatting around it in `\FN@mf@check` (hard-coded as `\textsuperscript`). `KOMA-Script` also stores plain content in `\multifootsep` and applies the formatting in the `check` function. `memoir`, however, stores the formatting directly in `\multifootsep` and applies no formatting later. Also, `latex-lab-footmisc.ltx`, in adding PDF tagging support for `footmisc` stores the tagging code directly in `\multifootsep`. I don’t know if there are others, but these cases are already relevant enough to spoil things. The problem is not that they disagree on the value of `\multifootsep`, but on the function(s) this macro is supposed to perform. That given, any other parties trying to partake in this feature have to handle things differently depending on the value of `\multifootsep`. All in all, `postnotes` takes the cautious stance of using internal variables, instead of the shared ones, to implement the `multiple` option. The only thing that really needs to be common is the value of the canceling kerns of `3sp` in `__postnotes_multiple_prepare:`.

```

381 \tl_new:N \l__postnotes_multisep_tl
382 \tl_const:Nn \c_postnotes_multi_notemarker_tl { 3sp }
383 \bool_new:N \l__postnotes_multiple_bool
384 \tl_new:N \l__postnotes_saved_spacefactor_multi_tl
385 \keys_define:nn { postnotes/setup }
386   {
387     multiple .bool_set:N = \l__postnotes_multiple_bool ,
388     multiple .default:n = true ,
389     multiple .initial:n = false ,
390     multisep .tl_set:N = \l__postnotes_multisep_tl ,
391     multisep .value_required:n = true ,
392     multisep .initial:n = {,} ,
393   }

```

I’m using the definitions in `latex-lab-footmisc.ltx` as base, see `texdoc latex-lab-footnotes`. For the formatting of the separator, though, I take inspiration from `KOMA-Script` and use `__postnotes_make_mark:nnn`, instead of hard-coding `\textsuperscript`.

```

394 \cs_new_protected:Npn \__postnotes_multiple_prepare:
395   {

```

```

396 \bool_if:NT \l__postnotes_multiple_bool
397 {
398 \kern -\c_postnotes_multi_notemarker_tl
399 \kern \c_postnotes_multi_notemarker_tl
400 \scan_stop:
401 }
402 }

```

Note that `__postnotes_multiple_check:` has to be called before any whatsits (labels, anchors, etc.) are placed, since they destroy `\lastkern` (see <https://chat.stackexchange.com/transcript/message/66554870#66554870>, thanks Ulrike Fischer).

```

403 \cs_new_protected:Npn \__postnotes_multiple_check:
404 {
405 \bool_if:NT \l__postnotes_multiple_bool
406 {
407 \dim_compare:nNnT
408 { \c_postnotes_multi_notemarker_tl } = { \lastkern }
409 {
410 \tl_set:Nc \l__postnotes_saved_spacefactor_multi_tl
411 { \int_use:N \spacefactor }
412 \unkern
413 \unkern
414 \tag_socket_use:n { postnotes/multisep/begin }
415 \__postnotes_make_mark:Vnn \l__postnotes_multisep_tl { } { }
416 \tag_socket_use:n { postnotes/multisep/end }
417 \spacefactor \l__postnotes_saved_spacefactor_multi_tl
418 \scan_stop:
419 }
420 }
421 }

```

sort option

```

422 \bool_new:N \l__postnotes_sort_bool
423 \keys_define:nn { postnotes/setup }
424 {
425 sort .bool_set:N = \l__postnotes_sort_bool ,
426 sort .initial:n = true ,
427 sort .default:n = true ,
428 }

```

checkduplicates and checkfloats options

```

429 \bool_new:N \l__postnotes_check_dupli_bool
430 \bool_new:N \l__postnotes_check_floats_bool
431 \keys_define:nn { postnotes/setup }
432 {
433 checkduplicates .bool_set:N = \l__postnotes_check_dupli_bool ,
434 checkduplicates .default:n = true ,
435 checkduplicates .initial:n = true ,
436 checkfloats .bool_set:N = \l__postnotes_check_floats_bool ,
437 checkfloats .default:n = true ,
438 checkfloats .initial:n = false ,
439 }

```

maybemulti option

```
440 \bool_new:N \l__postnotes_maybe_multi_bool
441 \keys_define:nn { postnotes/setup }
442 {
443   maybemulti .bool_set:N = \l__postnotes_maybe_multi_bool ,
444   maybemulti .default:n = true ,
445   maybemulti .initial:n = false ,
446 }
```

counteraux option

```
447 \bool_new:N \g__postnotes_counteraux_bool
448 \keys_define:nn { postnotes/setup }
449 {
450   counteraux .bool_gset:N = \g__postnotes_counteraux_bool ,
451   counteraux .default:n = true ,
452   counteraux .initial:n = false ,
453 }
```

```
454 \AddToHook { begindocument/before }
455 {
456   \bool_if:NT \g__postnotes_counteraux_bool
457     { \postnotessetup { sort=false } }
458   \keys_define:nn { postnotes/setup }
459     {
460       counteraux .code:n =
461         {
462           \msg_warning:nnn { postnotes }
463             { option-preamble-only } { counteraux }
464         } ,
465     }
466 }
```

```

\pnsetcounteraux
\pnaddtounteraux
\postnote@setcounteraux
\postnote@addtounteraux
467 \cs_new_protected:Npn \postnote@setcounteraux #1
468   { \int_gset:Nn \g__postnotes_postnote_counteraux_int { #1 } }
469 \cs_new_protected:Npn \postnote@addtounteraux #1
470   { \int_gadd:Nn \g__postnotes_postnote_counteraux_int { #1 } }
471 \NewDocumentCommand \pnsetcounteraux { m }
472 {
473   \@bsphack
474   \legacy_if:nT { @filesw }
475   {
476     \iow_shipout_e:Nn \@auxout
477       { \token_to_str:N \postnote@setcounteraux { #1 } }
478   }
479   \@esphack
480 }
481 \NewDocumentCommand \pnaddtounteraux { m }
482 {
483   \@bsphack
484   \legacy_if:nT { @filesw }
```

```

485     {
486       \iow_shipout_e:Nn \@auxout
487       { \token_to_str:N \postnote@addtocounteraux { #1 } }
488     }
489     \@esphack
490 }

```

(End of definition for `\pnsetcounteraux` and others.)

`\postnotesetup`

`\postnotesetup` Provide `\postnotesetup`.

```

\postnotesetup{options}

491 \NewDocumentCommand \postnotesetup { m }
492 { \keys_set:nn { postnotes/setup } {#1} }

```

(End of definition for `\postnotesetup`.)

4 `\postnote`

Different from the traditional `\footnotemark` / `\footnotetext` system, in the context of end notes, the functionality which corresponds to `\footnotetext` is simply to store the data to be typeset later. Hence, some of the problems that afflict footnotes do not apply to end notes. Namely, and as far as I can tell, they can be used in “inner horizontal mode” (`\mbox` etc.), and in math mode, and if the “text” will be typeset in the same page as the “mark” is of little concern.

However, the separation between “mark” and “text” is still useful in other contexts: floats and contexts where multiple typesetting passes are performed. David Carlisle and Ulrike Fischer shared some thoughts on the matter at the TeX.SX chat: <https://chat.stackexchange.com/transcript/message/60754383#60754383>.

The interesting questions here are: if they are replaceable in their roles in these contexts and how much would we lose by providing them. In analyzing this, we have to distinguish two situations: when `\footnotemark` is called with no argument (and thus steps the counter), and when it is called with the optional argument (and thus refrains from stepping the counter).

For floats, the problem they pose is that they may disturb the *ordering* of the notes. This particular issue can be solved by using `\footnotemark` without argument, and manually adjusting the counter on subsequent calls to `\footnotetext`. A good example of the technique is <https://tex.stackexchange.com/a/43694>. True, a user may wish to specify the mark explicitly, but doesn’t necessarily need to do it to solve the ordering issue.

Multiple typesetting passes of content are much harder. And they abound: the standard classes’ `\caption` typesets the caption once, if it is short, but twice if it is longer than a line; `amsmath`’s math environments perform a measuring pass before actually typesetting the equations; `amsmath`’s `\text` macro runs the contents through `\mathchoice` (which typesets the contents four times) when in math mode; `tabularx` and `tabularray` also perform measuring passes of their tables; so does `csquotes`’ blockquotes; and certainly

more that I'm unaware. A number of these places offer some one or another way to mitigate the issue: `amsmath`, `tabularx`, `csquotes` and (optionally) `tabularray` restore counter values after measuring steps; `amsmath` offers a boolean to indicate when it is a measuring pass; `csquotes` offers further handles. But the standard `\caption` offers none, and neither does `amsmath`'s `\text` macro. Well, the `pkgcaption` package can disable the multiple passes for `\caption` with the option `singlelinecheck`, but it is not reasonable to require it for our purposes, so we must assume the worst case.

Enrico Gregorio is categorical in stating that `\endnotemark` and `\endnotetext` are required for `enotez` to handle `\caption`, which apparently it didn't offer originally: "The package should implement `\endnotemark` and `\endnotetext` for this case. According to the documentation, the author deems them to not be needed: he's wrong." (<https://tex.stackexchange.com/a/314937>). See also <https://tex.stackexchange.com/a/43794> and <https://tex.stackexchange.com/a/358207>.

In this scenario, when there's no way around the multiple passes, `\footnotemark` can only handle the general case if used with an argument, precisely because it inhibits the stepping of the counter. Otherwise the counter is stepped multiple times, and we'd get the wrong number (and mark). In some circumstances, if we know the number of passes is deterministic, we might get away by adjusting the counter manually (`\caption` may be dealt with this way: if we know it to be two lines, we can decrement the counter before it and get correct results, even hyperlinked). But in cases which adjusting the counter is sufficient, end notes can be dealt with in the same way, and doesn't need the separation between "mark" and "text". So, what is distinctive of the kernel's footnote apparatus, which allows it as much flexibility as one would like, is receiving an arbitrary number as argument and not stepping the counter. And as far as `\footnotemark` and `\footnotetext` are concerned, the main point of the optional argument is really to "manually establish the relation" between the two of them. So, if *not stepping the counter* is what is needed to handle the general case, is it viable to do so? What would we lose in so doing?

When receiving an arbitrary number as argument, as the kernel functionality for footnotes and other endnotes packages do, this value is expected to be printed as such, hence it must correspond to the `postnote` counter (in our case). But this counter is in the hands of the user, and can be reset along the document, thus its uniqueness cannot be ensured. But not stepping `postnote` is perfectly viable, as it just aims at storing how the mark is to be typeset. However, not stepping the ID counter would complicate things considerably. Not doing so implies we'd lose the connection we have between the "mark" and the corresponding "text". We might add the "text" to the queue, but all the metadata would be lost, including the `hyperref` anchor, but really the set of data without which the kind of functionality offered would be nonviable, or severely hampered. Not stepping `postnote` but stepping the ID counter also is not sufficient, because we'd get a note in duplicity. We could naively think that a gap in the ID is not a problem, and just not add the duplicate to the queue. But how could we tell the difference between a legitimate and an illegitimate step of the ID counter?

I have not been able to devise a way to "reconnect" "text" and "mark" in the absence of the unique ID counter. The most promising idea was to have mandatory arguments to `\postnotemark` and `\postnotetext` receiving a `<label>` which we could use to identify their counterparts, but I was not able to go through with this, and the attempts all increased complexity considerably. It is not just a label/ref system, there's got to be a one-to-one correspondence between the sets, uniqueness has to be ensured on both sides, and there cannot be "lone" marks or texts (a bijection). Besides, this label based system of identification would have to live side-by-side with the one based on the

counter. So, even if we'd have unique IDs, we wouldn't know beforehand in what form it comes. Considering the ID is used to build the variable name in which we store the note's information, this would also complicate things.

Besides, there are ways to get things working with multiple passes without the “mark”/“text” partition. As mentioned, there are a number of cases which offer some kind of “handle” or way to identify the multiple passes. `csquotes` has a dedicated hook that can be used. `amsmath` sets the `measuring@` boolean (which `hyperref` also defines). So, not all cases are as tricky as `\caption` or `\text`, and even that can be decently dealt with without a separation between “mark” and “text”. Besides, in difficult cases, the package offers a `nomark` option to `\postnote` to place a note, but `typeset no mark`. Then we can `typeset` a mark with `\postnoteref` referring to a `\label` in the note of interest. This would result in a correct mark without duplicity, and in a correct link from there to the note's text at `\printpostnotes`. The drawback is that the placement of `\postnote` would be important, and results sensitive to it. All the metadata is collected at the point of `\postnote`, anchor included, not at the point of `\postnoteref`. So the consequences are a slightly off backlink, possibly imprecise metadata, etc. Considering `hyperref` itself shies away completely from linking `\footnotemark` with an argument, I'd say there's some gain.

The truth is there are some trade-offs, there's no “ideal” solution. Still, all in all, my judgment is that the unique ID counter is worth more than the inconveniences of an occasional `\postnote[nomark]` referenced with `\postnoteref`, and even that should not be needed much. So, for the time being, until something else shakes this balance, I won't be offering `\postnotemark` and `\postnotetext`.

For the `hyperref` support for cross-references in `\postnote`, I've moved back and forth quite a lot. One of the ideas I fancied was using `\refstepcounter` and let `hyperref` do its job. But, since I want to have control of the anchor/destination name on both “sides”, I'd have to set `\theHpostnote` locally before calling `\refstepcounter`, otherwise results might sensitive to user calls to `\counterwithin` (see <https://github.com/latex3/hyperref/issues/230>, thanks Ulrike Fischer). However, even if that worked well for the default case, we still had to setup things manually, in case of a manually supplied mark. All in all, I'm just calling `\stepcounter`, setting the relevant cross-reference variables once and setting the anchor manually.

`postnote` is the public, user facing, counter for `\postnote`. It determines how the note's mark gets to be `typeset`. It can be `reset`, `set`, and have its printed representation changed. Of course, whether those are meaningful is up to the user.

```
493 \newcounter { postnote }
```

```
\g__postnotes_note_id_int \g__postnotes_note_id_int is the internal, unique counter which provides the ID number
\l_postnotes_note_id_tl of each note. It ties “mark” and “text” together, is also the connection between each
\g__postnotes_queue_seq note and its data, including the content, which is stored in a property list named according
\l_postnotes_counteraux_step_int to \__postnotes_data_name:n and the ID number. \l_postnotes_note_id_tl is a
\l_postnotes_mark_typeset_tl convenience variable storing the counter's value. \g__postnotes_queue_seq stores the
\l_postnotes_note_set_labels_tl sequence of notes' IDs to be processed by the next call of \printpostnotes.
```

```
494 \int_new:N \g__postnotes_note_id_int
495 \tl_new:N \l_postnotes_note_id_tl
496 \tl_set:Nn \l_postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
497 \seq_new:N \g__postnotes_queue_seq
498 \int_new:N \l__postnotes_counteraux_step_int
```

```

499 \tl_new:N \l__postnotes_mark_typeset_tl
500 \tl_new:N \l__postnotes_note_set_labels_tl

```

(End of definition for `\g__postnotes_note_id_int` and others.)

`\postnote` Provide `\postnote`.

```

\postnote[options]{note text}

501 \NewDocumentCommand \postnote { 0 { } +m }
502 { \__postnotes_note:nn {#1} {#2} }

```

(End of definition for `\postnote`.)

`__postnotes_note:nn` The internal version of `\postnote`. The `postnotes/note/begin` hook is meant to provide a place from where some additional setup for the note can be performed. This is being used for adding support for some features/packages, but can also be used, for example, to add an extra local property for `zref`.

```

\__postnotes_note:nn [options] {note content}

503 \NewHook { postnotes/note/begin }
504 \cs_new_protected:Npn \__postnotes_note:nn #1#2
505 {
506   \group_begin:
507   \keys_set:nn { postnotes/note } {#1}
508   \bool_if:NT \l__postnotes_nomark_bool { \@bsphack }
509   \__postnotes_inhibit_note:F
510   {
511     \int_gincr:N \g__postnotes_note_id_int
512     \tl_if_empty:NTF \l__postnotes_mark_tl
513     {
514       \stepcounter { postnote }
515       \int_set:Nn \l__postnotes_counteraux_step_int { 1 }
516       \bool_if:NT \g__postnotes_counteraux_bool
517       {
518         \exp_args:NNe \prop_gpop:NnNT \g__postnotes_counteraux_prop
519         { \l__postnotes_note_id_tl } \l__postnotes_tmpa_tl
520         { \int_set:Nn \c@postnote { \l__postnotes_tmpa_tl } }
521         \tl_clear:N \l__postnotes_tmpa_tl
522       }
523       \tl_set:Ne \l__postnotes_mark_tl { \thepostnote }
524     }
525     { \int_set:Nn \l__postnotes_counteraux_step_int { 0 } }
526   \UseHook { postnotes/note/begin }
527   \seq_gput_right:Ne \g__postnotes_queue_seq
528   { \l__postnotes_note_id_tl }
529   \cs_set:Npn \@currentcounter { postnote }
530   \cs_set:Npe \@currentlabel { \p@postnote \l__postnotes_mark_tl }
531   \__postnotes_store:nn { \l__postnotes_note_id_tl } {#2}
532   \tl_set_eq:NN \l__postnotes_mark_typeset_tl \l__postnotes_mark_tl

```

Prefer label for typesetting measuring passes, if available, see comments at `__postnotes_inhibit_note:F`.

```

533     \bool_lazy_or:nnT
534     { \g__postnotes_countersaux_bool }
535     { \l__postnotes_maybe_multi_bool }
536     {
537       \bool_lazy_and:nnT
538       { ! \g__postnotes_firstrun_bool }
539       {
540         ! \cs_if_exist_p:c
541           { \c__postnotes_ref_prefix_tl @ mark @ \l_postnotes_note_id_tl }
542         }
543       { \__postnotes_get_label_if_exist:N \l__postnotes_mark_typeset_tl }
544     }
545     \tl_set:Nn \l__postnotes_note_set_labels_tl
546     {
547       \MakeLinkTarget* { postnote. \l_postnotes_note_id_tl .mark }
548       \__postnotes_set_mark_page_label:ee { \l_postnotes_note_id_tl }
549       { \int_use:N \l__postnotes_countersaux_step_int }
550       \__postnotes_set_user_labels:
551     }
552     \bool_if:NTF \l__postnotes_nomark_bool
553     {
554       \tag_socket_use:n { postnotes/nomark/begin }
555       \l__postnotes_note_set_labels_tl
556       \tag_socket_use:n { postnotes/nomark/end }
557     }
558     {
559       \__postnotes_typeset_mark:eVN
560       { \l_postnotes_note_id_tl } \l__postnotes_mark_typeset_tl
561       \l__postnotes_note_set_labels_tl
562     }
563   }
564   \bool_if:NT \l__postnotes_nomark_bool { \@esphack }
565   \group_end:
566 }

```

(End of definition for `__postnotes_note:nn`.)

Options for `\postnote`.

```

567 \tl_new:N \l__postnotes_mark_tl
568 \bool_new:N \l__postnotes_nomark_bool
569 \fp_new:N \l__postnotes_sort_num_fp
570 \str_new:N \l__postnotes_note_label_str
571 \bool_new:N \l__postnotes_manual_sortnum_bool
572 \keys_define:nn { postnotes/note }
573 {
574   markstr .tl_set:N = \l__postnotes_mark_tl ,
575   markstr .value_required:n = true ,
576   sortnum .code:n =
577   {
578     \fp_set:Nn \l__postnotes_sort_num_fp {#1}
579     \bool_set_true:N \l__postnotes_manual_sortnum_bool
580   } ,

```

```

581     sortnum .value_required:n = true ,
582     mark .meta:n =
583     {
584         markstr = {#1} ,
585         sortnum = {#1} ,
586     } ,
587     mark .value_required:n = true ,
588     nomark .bool_set:N = \l__postnotes_nomark_bool ,
589     nomark .default:n = true ,
590     label .str_set:N = \l__postnotes_note_label_str ,
591     label .value_required:n = true ,
592     maybemulti .bool_set:N = \l__postnotes_maybe_multi_bool ,
593     maybemulti .default:n = true ,
594 }

```

`__postnotes_inhibit_note:TF` In contexts of multiple passes of content, it may be needed, or preferred, to inhibit the note altogether to avoid side effects and duplicity. This conditional, obviously, will always return the true branch unless something is done in the `postnotes/note/inhibit` hook. This hook is meant to handle support for packages or features which may justify note inhibition, and the code there should set `\l__postnotes_inhibit_note_bool`, `\l__postnotes_print_plain_mark_bool` and `\l__postnotes_print_plain_mark_stepcounter_bool` as appropriate to the case.

```

595 \bool_new:N \l__postnotes_inhibit_note_bool
596 \bool_new:N \l__postnotes_print_plain_mark_bool
597 \bool_new:N \l__postnotes_print_plain_mark_stepcounter_bool
598 \NewHook { postnotes/note/inhibit }
599 \prg_new_protected_conditional:Npnn \__postnotes_inhibit_note: { F }
600 {
601     \bool_set_false:N \l__postnotes_inhibit_note_bool
602     \bool_set_false:N \l__postnotes_print_plain_mark_bool
603     \bool_set_false:N \l__postnotes_print_plain_mark_stepcounter_bool
604     \UseHook { postnotes/note/inhibit }

```

Printing a plain mark here may be needed because, if we are inhibiting the note in a “measuring context” and omit it completely, the measuring being performed will be off by the size of the mark. So, to ensure the measuring can be done correctly, we place the mark. What to do with the counter itself, depends on the situation. In places that are known to restore the counter values after the measuring pass, we can let the counter be stepped. And, actually we should do so, for example, in a `tabularx` with multiple postnotes, if we don’t step the counter, all the measuring will be done with the number of the first note. Otherwise, we don’t actually step the counter but, to typeset correctly the mark that would be printed if the counter had been stepped, we increment `\c@postnote` locally and grouped, and smuggle `\thepostnote` out of the group.

```

605     \bool_lazy_all:nT
606     {
607         { \l__postnotes_inhibit_note_bool }
608         { \l__postnotes_print_plain_mark_bool }
609         { ! \l__postnotes_nomark_bool }
610     }
611     {
612         \tl_if_empty:NTF \l__postnotes_mark_tl
613         {
614             \bool_if:NTF \l__postnotes_print_plain_mark_stepcounter_bool

```

```

615     {
616       \stepcounter { postnote }
617       \tl_set:Nc \l__postnotes_mark_typeset_tl { \thepostnote }
618     }
619     {
620       \group_begin:
621         \int_incr:N \c@postnote
622         \exp_args:NNNe
623         \group_end:
624         \tl_set:Nn \l__postnotes_mark_typeset_tl { \thepostnote }
625     }

```

If the note has a `label`, use a cross-reference to that as the mark instead. In principle, the procedure above is expected to work reasonably well for cases where we know whether we are in a measuring pass or not, and how it handles the counters (if it restores counters or not). This is true though, only if we are going in the expansion order of the document. If we are using the `counteraux` option, the mere sequence of the notes is no longer an indicator of who is a measuring pass of who, indeed the measuring pass does not even get to the `.aux` file. If the label exists, though, it is *known to be right* regardless of the case, since it belongs to the pass which actually gets typeset. Hence, if we have a label, it is more general and more reliable: use it.

```

626     \__postnotes_get_label_if_exist:N \l__postnotes_mark_typeset_tl
627   }
628   { \tl_set_eq:NN \l__postnotes_mark_typeset_tl \l__postnotes_mark_tl }
629   \group_begin:
630     \socket_assign_plug:nn { tagssupport/postnotes/multisep/begin } { noop }
631     \socket_assign_plug:nn { tagssupport/postnotes/multisep/end } { noop }
632     \__postnotes_typeset_mark_wrapper:nnn
633     { \__postnotes_make_mark:Vnn \l__postnotes_mark_typeset_tl { } { } }
634     { } { }
635   \group_end:
636 }
637 \bool_if:NTF \l__postnotes_inhibit_note_bool
638 { \prg_return_true: }
639 { \prg_return_false: }
640 }

```

(End of definition for `__postnotes_inhibit_note:TF`.)

```

\__postnotes_get_label_if_exist:N     \__postnotes_get_label_if_exist:N <\l__postnotes_mark_tl>
641 \cs_new_protected:Npn \__postnotes_get_label_if_exist:N #1
642 {
643   \str_if_empty:NTF \l__postnotes_note_label_str
644   {
645     \str_if_empty:NF \l__postnotes_note_zlabel_str
646     {
647       \exp_args:NV \zref@ifrefundefined \l__postnotes_note_zlabel_str
648       { }
649       {
650         \exp_args:NV \zref@ifrefcontainsprop
651         \l__postnotes_note_zlabel_str
652         { postnote@mark }
653         {

```

```

654         \exp_args:NNNo \exp_args:NNo \tl_set:Nn #1
655         {
656             \zref@extract
657             { \l__postnotes_note_zlabel_str } { postnote@mark }
658         }
659     }
660     { }
661 }
662 }
663 }
664 {
665     \exp_args:Ne \property_if_recorded:nnT
666     { postnotes@ \l__postnotes_note_label_str }
667     { postnotes/mark }
668     {
669         \tl_set:Ne #1
670         {
671             \property_ref:ee
672             { __postnotes_ \l__postnotes_note_label_str } { postnotes/mark }
673         }
674     }
675 }
676 }

```

(End of definition for `__postnotes_get_label_if_exist:N`.)

`__postnotes_typeset_mark:nnN` Auxiliary functions for mark typesetting in `__postnotes_note:nn`. `__postnotes_`
`__postnotes_typeset_mark_wrapper:nnn` `typeset_mark_wrapper:nnn` is based on the definition of `\@footnotemark` in the kernel.

```

        \__postnotes_typeset_mark:nnN {<note id>} {<mark>} {<labels>}
        \__postnotes_typeset_mark_wrapper:nnn {<mark>}
        {(begin tagging)} {(end tagging)}
677 \cs_new_protected:Npn \__postnotes_typeset_mark:nnN #1#2#3
678 {
679     \__postnotes_typeset_mark_wrapper:nnn
680     {
681         #3
682         \bool_if:NTF \l__postnotes_hyperlink_bool
683         {
684             \__postnotes_make_mark:nnn {#2}
685             { \hyper@linkstart { link } { postnote. #1 .text } }
686             { \hyper@linkend }
687         }
688         { \__postnotes_make_mark:nnn {#2} { } { } }
689     }
690     { \tag_socket_use:n { postnotes/mark/begin } }
691     { \tag_socket_use:n { postnotes/mark/end } }
692 }
693 \cs_generate_variant:Nn \__postnotes_typeset_mark:nnN { eVN }
694 \tl_new:N \l__postnotes_saved_spacefactor_tl
695 \cs_new_protected:Npn \__postnotes_typeset_mark_wrapper:nnn #1#2#3
696 {
697     \mode_leave_vertical:
698     \mode_if_horizontal:T

```

```

699     {
700       \tl_set:Ne \l__postnotes_saved_spacefactor_tl
701       { \int_use:N \spacefactor }
702       \__postnotes_multiple_check:
703       \nobreak
704     }
705     #2 % begin tagging socket
706     #1 % mark
707     #3 % end tagging socket
708     \__postnotes_multiple_prepare:
709     \mode_if_horizontal:T
710     { \spacefactor \l__postnotes_saved_spacefactor_tl }
711     \scan_stop:
712   }

```

(End of definition for __postnotes_typeset_mark:nnN and __postnotes_typeset_mark_wrapper:nnn.)

`__postnotes_set_user_labels:` Auxiliary function for user label setting in `__postnotes_note:nn`. Supports the `label` and `zlabel` options of `\postnote`.

```

713 \cs_new_protected:Npn \__postnotes_set_user_labels:
714   {
715     \str_if_empty:NF \l__postnotes_note_label_str
716     {
717       \exp_args:NV \label \l__postnotes_note_label_str
718       \property_record:ee { __postnotes_ \l__postnotes_note_label_str }
719       { postnotes/mark }
720     }
721     \str_if_empty:NF \l__postnotes_note_zlabel_str
722     { \exp_args:NV \zlabel \l__postnotes_note_zlabel_str }
723   }
724 \property_new:nnnn { postnotes/mark } { now } { } { \l__postnotes_mark_tl }

```

(End of definition for __postnotes_set_user_labels:.)

5 \postnoteref

`\postnoteref` Provide `\postnoteref`.

```

\postnoteref(*){\label}

725 \NewDocumentCommand \postnoteref { s m }
726   { \__postnotes_note_ref:nn {#1} {#2} }

```

(End of definition for \postnoteref.)

`__postnotes_note_ref:nn` The internal version of `\postnoteref`.

```

\__postnotes_note_ref:nn {\star bool} {\label}

```

```

727 \tl_new:N \l__postnotes_note_ref_label_tl
728 \cs_new_protected:Npn \__postnotes_note_ref:nn #1#2
729 {
730   \group_begin:
731     \tl_set:Nn \l__postnotes_note_ref_label_tl {#2}
732     \__postnotes_typeset_mark_wrapper:nnn
733     {
734       \bool_lazy_and:nnTF
735       { ! #1 }
736       { \l__postnotes_hyperlink_bool }
737       {
738         \hyperref [#2]
739         { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
740       }
741       { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
742     }
743     { \tag_socket_use:n { postnotes/postnoteref/begin } }
744     { \tag_socket_use:n { postnotes/postnoteref/end } }
745   \group_end:
746 }

```

(End of definition for __postnotes_note_ref:nn.)

6 \postnotesection

\postnotesection Provide \postnotesection.

```

\postnotesection[<options>]{<section content>}
747 \NewDocumentCommand \postnotesection { 0 { } +m }
748 {
749   \@bsphack
750   \__postnotes_section:nn {#1} {#2}
751   \@esphack
752 }

```

(End of definition for \postnotesection.)

__postnotes_section:nn The internal version of \postnotesection.

```

\__postnotes_section:nn {<options>} {<content>}
753 \int_new:N \g__postnotes_sectid_int
754 \cs_new_protected:Npn \__postnotes_section:nn #1#2
755 {
756   \group_begin:
757     \int_gincr:N \g__postnotes_sectid_int
758     \int_gincr:N \g__postnotes_note_id_int
759     \seq_gput_right:Ne \g__postnotes_queue_seq { \l__postnotes_note_id_tl }
760     \tl_gclear:N \g__postnotes_section_name_tl
761     \keys_set:nn { postnotes/section } {#1}
762     \__postnotes_set_section_page_label:e { \l__postnotes_note_id_tl }
763     \bool_if:NTF \l__postnotes_section_exp_bool

```



```

764         { \l_postnotes_store_section:ne { \l_postnotes_note_id_tl } {#2} }
765         { \l_postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
766     \group_end:
767 }

```

(End of definition for `\l_postnotes_section:nn`.)

Options for `\postnotessection`. Actually, I would have preferred to use “label” for the name option, but I feared I might need it further down the road for the traditional meaning.

```

768 \tl_new:N \g__postnotes_section_name_tl
769 \bool_new:N \l__postnotes_section_exp_bool
770 \keys_define:nn { postnotes/section }
771 {
772     name .tl_gset:N = \g__postnotes_section_name_tl ,
773     name .value_required:n = true ,
774     exp .bool_set:N = \l__postnotes_section_exp_bool ,
775     exp .initial:n = false ,
776     exp .default:n = true ,
777 }

```

7 `\printpostnotes`

`\printpostnotes` Provide `\printpostnotes`.

`\printpostnotes`

```

778 \NewDocumentCommand \printpostnotes { }
779 { \l__postnotes_print_notes: }

```

(End of definition for `\printpostnotes`.)

`\pnthechapter` User facing variables, aimed at making available some of the notes’ and sections’ metadata for the user at specific contexts.

```

\pnthechapternextnote 780 \tl_new:N \pnthechapter
\pnthesectionnextnote 781 \tl_new:N \pnthesection
\pnthepage 782 \tl_new:N \pnidnextnote
\pnidnextnote 783 \tl_new:N \pnthechapternextnote
784 \tl_new:N \pnthesectionnextnote
785 \tl_new:N \pnthepage

```

(End of definition for `\pnthechapter` and others.)

`\g__postnotes_print_postnotes_int` Auxiliary variables for `\l__postnotes_print_notes::`

```

\g__postnotes_print_queue_seq 786 \int_new:N \g__postnotes_print_postnotes_int
\l_postnotes_print_note_id_tl 787 \seq_new:N \g__postnotes_print_queue_seq
\l__postnotes_print_note_id_next_tl 788 \tl_new:N \l_postnotes_print_note_id_tl
\l_postnotes_print_counter_tl 789 \tl_new:N \l__postnotes_print_note_id_next_tl
\l__postnotes_print_mark_tl 790 \tl_new:N \l__postnotes_print_counter_tl
\l_postnotes_print_typeset_mark_tl 791 \tl_new:N \l__postnotes_print_mark_tl
\l_postnotes_print_type_curr_tl 792 \tl_new:N \l__postnotes_print_typeset_mark_tl
\l_postnotes_print_type_next_tl 793 \tl_new:N \l__postnotes_print_type_curr_tl
\l_postnotes_print_type_prev_tl 794 \tl_new:N \l__postnotes_print_type_next_tl
\l_postnotes_print_content_tl
\l_postnotes_clear_queue_seq

```

```

795 \tl_new:N \l__postnotes_print_type_prev_tl
796 \tl_new:N \l__postnotes_print_content_tl
797 \seq_new:N \l__postnotes_clear_queue_seq

```

(End of definition for `\g__postnotes_print_postnotes_int` and others.)

`__postnotes_print_notes:` hooks. Both meant at providing points of entry for additional setup, specially to add support to packages and features which require it. The `postnotes/print/begin` hook is run early in `__postnotes_print_notes:` and only once per call, after the user options have been processed. The `postnotes/print/note/begin` hook is run once for each note, at the point where environment variables are being set or restored, before the typesetting of either the mark or the text, but within a group of its own of each note.

```

798 \NewHook { postnotes/print/begin }
799 \NewHook { postnotes/print/note/begin }

```

The `postnotetext` is a counter used to restore the original value of `postnote` at the time of printing, for the purposes of cross-referencing, it should be different from `postnote` if a note may occur inside `\printpostnotes`. The `postnotesection` is a counter which is stepped for every postnote section which gets to be actually typeset. It's aim is to provide a valid "enclosing counter" to `postnote` in the context of `\printpostnotes`. Since we don't know where `postnote` may have been reset along the document, in the general case, we can't rely on any other preexisting counter. This means that the particular value of `postnotesection` is of little practical meaning, it really is just meant to provide recognizable "bounds" for `postnote` along the printing of the notes. Indeed, it is initialized to a very high value (larger than the conceivable number of postnote sections in a document), so that "marks" and "texts" don't mix in the same reference list, which would occur if the enclosing counters of both belonged to the same range, and with somewhat arbitrary results, since we cannot ensure the step of the enclosing counter along the document matches `postnotesection`. This is actually a tricky problem from the cross-referencing standpoint: two different things, which should be of the same type, are reset along the document, but shouldn't really be mixed together. They are both $\text{\LaTeX} 2_{\epsilon}$ counters, since they may be required externally. Their main intended use case is to support `zref-clever`, but in principle they can be of general use.

```

800 \newcounter { postnotetext }
801 \newcounter { postnotesection }
802 \setcounter { postnotesection } { 10000 }

```

`__postnotes_print_notes:` The internal version of `\printpostnotes`.

```

\__postnotes_print_notes:
803 \cs_new_protected:Npn \__postnotes_print_notes:
804 {
805   \group_begin:
806     \int_gincr:N \g__postnotes_print_postnotes_int
807     \__postnotes_split_labelseq:

```

We make the cut at this point. `\g__postnotes_print_queue_seq` is stored won't receive any more notes for the duration of this call of `\printpostnotes`, any notes added to the queue from this point on belong to the next call of `\printpostnotes`.

```

808     \bool_lazy_and:nnTF

```

```

809     { \g__postnotes_countersaux_bool }
810     { ! \g__postnotes_firstrun_bool }
811     {
812         \seq_gset_eq:NN \g__postnotes_print_queue_seq
813         \g__postnotes_print_labelseq_queue_seq
814     }
815     {
816         \seq_gset_eq:NN \g__postnotes_print_queue_seq
817         \g__postnotes_queue_seq
818     }
819     \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_print_queue_seq
820     \seq_gclear:N \g__postnotes_queue_seq

```

The purpose of this label is to provide a point for splitting the labelseq with the countersaux option. It only needs to exist, it doesn't even store the page value. The `__postnotes_set_print_page_label:e` done at `__postnotes_set_headers_vars_first:` right below does not suffice for this purpose for two reasons. It won't be set if the queue is empty (or not yet populated), and also it comes after the heading (as it must), which means `\postnotessections` added through hooks to it will come before it.

```

821     \__postnotes_set_pre_print_label:e
822     { \int_use:N \g__postnotes_print_postnotes_int }
823     \seq_if_empty:NTF \g__postnotes_print_queue_seq
824     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
825     {
826         \pnheading
827         \UseHook { postnotes/print/begin }
828         \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
829         \__postnotes_check_duplicates:N \g__postnotes_print_queue_seq
830         \__postnotes_sort_queue:N \g__postnotes_print_queue_seq
831         \__postnotes_check_floats:N \g__postnotes_print_queue_seq
832         \bool_gset_true:N \g__postnotes_header_vars_next_bool
833         \__postnotes_get_headers_data:N \g__postnotes_print_queue_seq
834         \__postnotes_set_headers_vars_first:

```

Ensure the first note after a heading has paragraph indentation when `listenv` is `none`. `endnotes` uses a workaroundsish solution in `\noteheading`, setting a box and then skipping back a line. Enrico Gregorio is correct, though, in criticizing it at https://tex.stackexchange.com/q/575905#comment1450213_575915, and suggests the use of `\@afterindenttrue`, which is what `indentfirst` does (we do the same, just locally).

```

835         \bool_if:NF \l__postnotes_print_as_list_bool
836         {
837             \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
838             \@afterindenttrue
839         }
840     \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_print_queue_seq }
841     {
842         \seq_gpop_left:NN \g__postnotes_print_queue_seq
843         \l_postnotes_print_note_id_tl
844         \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
845         { type } \l__postnotes_print_type_curr_tl
846         \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
847         { % type_curr = 'section'
848             \seq_if_empty:NTF \g__postnotes_print_queue_seq
849             {

```

```

850         \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
851         \tl_set:Nn \l__postnotes_print_type_next_tl { close }
852     }
853     {
854         \seq_get_left:NN \g__postnotes_print_queue_seq
855         \l__postnotes_print_note_id_next_tl
856         \__postnotes_prop_get:nnN
857         { \l__postnotes_print_note_id_next_tl }
858         { type } \l__postnotes_print_type_next_tl
859     }

```

We only process the entry if `type_next` is `note`: here are skipped empty sections.

```

860         \tl_if_eq:NnT \l__postnotes_print_type_next_tl { note }
861     {
862         \stepcounter { postnotessection }
863         \group_begin:
864             \__postnotes_prop_get:nnN
865             { \l__postnotes_print_note_id_tl }
866             { thechapter } \pnthechapter
867             \__postnotes_prop_get:nnN
868             { \l__postnotes_print_note_id_tl }
869             { thesection } \pnthesection
870             \tl_set:NV \pnidnextnote
871             \l__postnotes_print_note_id_next_tl
872             \__postnotes_prop_get:nnN
873             { \l__postnotes_print_note_id_next_tl }
874             { thechapter } \pnthechapternextnote
875             \__postnotes_prop_get:nnN
876             { \l__postnotes_print_note_id_next_tl }
877             { thesection } \pnthesectionnextnote
878             \__postnotes_prop_get:nnN
879             { \l__postnotes_print_note_id_tl }
880             { content } \l__postnotes_print_content_tl
881             \l__postnotes_print_content_tl
882         \group_end:

```

Set `type_prev` for the next iteration.

```

883             \tl_set:NV \l__postnotes_print_type_prev_tl
884             \l__postnotes_print_type_curr_tl
885         }
886     }
887     { % type_curr = 'note'
888         \seq_if_empty:NTF \g__postnotes_print_queue_seq
889         {
890             \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
891             \tl_set:Nn \l__postnotes_print_type_next_tl { close }
892         }
893         {
894             \seq_get_left:NN \g__postnotes_print_queue_seq
895             \l__postnotes_print_note_id_next_tl
896             \__postnotes_prop_get:nnN
897             { \l__postnotes_print_note_id_next_tl }
898             { type } \l__postnotes_print_type_next_tl
899         }
900         \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }

```

```

901     {
902         \bool_if:NTF \l__postnotes_print_as_list_bool
903         { \exp_args:NV \begin \l__postnotes_print_env_tl }
904         { \group_begin: }
905         \tag_socket_use:n { postnotes/printlist/begin }
906         \l__postnotes_print_format_tl
907     }
908 \group_begin:
909 \UseHook { postnotes/print/note/begin }
910 \__postnotes_get_pageref:Ne \pnthepage
911   { mark@ \l__postnotes_print_note_id_tl }
912 \__postnotes_prop_get:nnN
913   { \l__postnotes_print_note_id_tl }
914   { mark } \l__postnotes_print_mark_tl
915 \__postnotes_prop_get:nnN
916   { \l__postnotes_print_note_id_tl }
917   { counter } \l__postnotes_print_counter_tl
918 \__postnotes_prop_get:nnN
919   { \l__postnotes_print_note_id_tl }
920   { content } \l__postnotes_print_content_tl
921 \cs_set:Npn \@currentcounter { postnotetext }
922 \int_set:Nn \c@postnotetext
923   { \l__postnotes_print_counter_tl }
924 \cs_set:Npe \@currentlabel
925   { \p@postnote \l__postnotes_print_mark_tl }
926 \tl_set:Nn \l__postnotes_print_typeset_mark_tl
927   {
928     \tag_socket_use:n { postnotes/printmark/begin }
929     \MakeLinkTarget*
930     { postnote. \l__postnotes_print_note_id_tl .text }
931     \__postnotes_set_text_page_label:e
932     { \l__postnotes_print_note_id_tl }
933     \l__postnotes_pre_textmark_tl
934     \__postnotes_typeset_text_mark:eV
935     { \l__postnotes_print_note_id_tl }
936     \l__postnotes_print_mark_tl
937     \l__postnotes_post_textmark_tl
938     \tag_socket_use:n { postnotes/printmark/end }
939   }

```

Note that the placement of the tagging socket for the list case may depend on the tagging structure, in other words, on the content of the socket. It currently does nothing for the list case, so I've placed it in the "potentially most useful place". Review this if the content changes. Leave vertical mode after `\item` for the list case to avoid "perhaps a missing `\item`" error for empty notes (see `pn-bug-empty-postnote01.lvt`). And leave vertical mode before the note (and the tagging socket) for `listenv=none` (see <https://github.com/gusbrs/postnotes/issues/8#issuecomment-2429501962>, thanks Ulrike Fischer).

```

940     \bool_if:NTF \l__postnotes_print_as_list_bool
941     {
942         \item
943         [
944             \tag_socket_use:n { postnotes/printnote/begin }
945             \l__postnotes_print_typeset_mark_tl
946         ]

```

```

947         \mode_leave_vertical:
948     }
949     {
950         \mode_leave_vertical:
951         \tag_socket_use:n { postnotes/printnote/begin }
952         \l__postnotes_print_typeset_mark_tl
953     }
954     \tag_socket_use:n { postnotes/printtext/begin }
955     \l__postnotes_print_content_tl
956     \tag_socket_use:n { postnotes/printtext/end }
957     \tag_socket_use:n { postnotes/printnote/end }
958     \l__postnotes_post_printnote_tl
959     \group_end:
960     \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
961     {
962         \tag_socket_use:n { postnotes/printlist/end }

```

Ensure `\par` at the end of `\printpostnotes` (see <https://github.com/u-fischer/tagpdf/issues/68#issuecomment-1587343876>, thanks Ulrike Fischer).

```

963         \bool_if:NTF \l__postnotes_print_as_list_bool
964         {
965             \exp_args:NV \end \l__postnotes_print_env_tl
966             \par
967         }
968         {
969             \par
970             \group_end:
971         }
972     }

```

Set `type_prev` for the next iteration.

```

973         \tl_set:NV \l__postnotes_print_type_prev_tl
974         \l__postnotes_print_type_curr_tl
975     }
976 }
977 \AddToHookNext { shipout/after }
978 { \bool_gset_false:N \g__postnotes_header_vars_next_bool }

```

We won't use the variables anymore, clear them to reduce memory usage.

```

979     \seq_map_inline:Nn \l__postnotes_clear_queue_seq
980     { \__postnotes_prop_gclear:n { ##1 } }
981 }
982 \group_end:
983 }

```

(End of definition for `__postnotes_print_notes:.`)

```

984 \msg_new:nnn { postnotes } { empty-printpostnotes }
985 { Empty~'\iow_char:N\printpostnotes'~\msg_line_context:. }

```

`__postnotes_typeset_text_mark:nn` Auxiliary function for mark typesetting in `__postnotes_print_notes:.`

```

\__postnotes_typeset_text_mark:nn {<note id>} {<mark>}

```

```

986 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
987 {
988   \bool_lazy_and:nnTF
989     { \l__postnotes_hyperlink_bool }
990     { \l__postnotes_backlink_bool }
991     {
992       \__postnotes_make_text_mark:nnn {#2}
993       { \hyper@linkstart { link } { postnote. #1 .mark } }
994       { \hyper@linkend }
995     }
996     { \__postnotes_make_text_mark:nnn {#2} { } { } }
997 }
998 \cs_generate_variant:Nn \__postnotes_typeset_text_mark:nn { eV }

```

(End of definition for `__postnotes_typeset_text_mark:nn`.)

Print auxiliary

The conditions used to split `\g__postnotes_labelseq_seq` are subtly different depending on whether we are using `\g__postnotes_countersaux_bool` or not. In the standard case, the numbering of the floats are set at “expansion time”, so they belong where they are set. With `counteraux`, the numbering of floats are set at “shipout time”, so they belong where they are typeset. In other words, with `counteraux` notes on floats can cross the boundaries of `\printpostnotes`, while without it, they must not. Furthermore, the purpose of `\g__postnotes_labelseq_seq` is different in each case. With `counteraux` it is used to build the actual print queue, while in the standard case it is only used in `__postnotes_check_floats:N`. Therefore, with `counteraux` the cut is made at the place the preprint label for the current `\printpostnotes` is found, while in the standard case, `\g__postnotes_note_id_int` is used directly to distribute the elements between the current `\printpostnotes` and future ones.

`__postnotes_split_labelseq:`

```

999 \seq_new:N \g__postnotes_print_labelseq_queue_seq
1000 \cs_new_protected:Npn \__postnotes_split_labelseq:
1001 {
1002   \group_begin:
1003     \seq_clear:N \l__postnotes_tmpa_seq
1004     \seq_clear:N \l__postnotes_tmpb_seq
1005     \bool_if:NTF \g__postnotes_countersaux_bool
1006     {
1007       \tl_set:Nc \l__postnotes_tmpa_tl
1008       { { preprint } { \int_use:N \g__postnotes_print_postnotes_int } }

```

The preprint label of a `\printpostnotes` at the end of the document may not have been written: if it’s empty, it may not be shipped out at all. But, since it’s a counter, stepped sequentially and not floating, even if it is transitorily missing, it will be at the end. In other words, if this one is not found, there will be no subsequent preprints in the sequence.

```

1009     \seq_if_in:NVF \g__postnotes_labelseq_seq \l__postnotes_tmpa_tl
1010     { \seq_gput_right:NV \g__postnotes_labelseq_seq \l__postnotes_tmpa_tl }
1011     \bool_do_until:nn
1012     { \tl_if_eq_p:NN \l__postnotes_tmpb_tl \l__postnotes_tmpa_tl }
1013     {

```

```

1014         \seq_gpop_left:NN \g__postnotes_labelseq_seq \l__postnotes_tmpb_tl
1015         \str_case:enT
1016         { \tl_item:Nn \l__postnotes_tmpb_tl { 1 } }
1017         {
1018             { mark } { }
1019             { section } { }
1020         }
1021         {

```

If the id of the labelseq item is larger than the current note id, we don't have data for the note at this point, and cannot print it. Period. Now, usually this will occur due to transitory effects. But it is theoretically possible that a float is sent to the top of the page and gets typeset before a "future `\printpostnotes`". So what we cannot print, we give back to the label sequence of the next `\printpostnotes`. If they are transitory, they will eventually go away. If they are not, better to keep them with the wrong numbering than dropping it altogether. Alas, there's nothing else we could do, short of writing the whole data to the `.aux` file, which is clearly not worth this corner case.

```

1022             \int_compare:nNnTF
1023             { \tl_item:Nn \l__postnotes_tmpb_tl { 2 } }
1024             >
1025             { \g__postnotes_note_id_int }
1026             {
1027                 \seq_put_right:Ne \l__postnotes_tmpb_seq
1028                 \l__postnotes_tmpb_tl
1029             }
1030             {
1031                 \seq_put_right:Ne \l__postnotes_tmpa_seq
1032                 { \tl_item:Nn \l__postnotes_tmpb_tl { 2 } }
1033             }
1034         }

```

No need for the F branch of `\str_case:enT`, at this point the preprint of past `\printpostnotes` can no longer be here, and we don't go further than the current one. In theory, we could even go without `\str_case:enT`, but let's play safe and keep the function robust against future changes of the code.

```

1035         }
1036         \seq_gset_eq:NN \g__postnotes_print_labelseq_queue_seq
1037         \l__postnotes_tmpa_seq
1038         \seq_concat:NNN \l__postnotes_tmpa_seq \l__postnotes_tmpb_seq
1039         \g__postnotes_labelseq_seq
1040         \seq_gset_eq:NN \g__postnotes_labelseq_seq \l__postnotes_tmpa_seq
1041     }
1042     {
1043         \seq_map_inline:Nn \g__postnotes_labelseq_seq
1044         {
1045             \str_case:enT
1046             { \tl_item:nn { ##1 } { 1 } }
1047             {
1048                 { mark } { }
1049                 { section } { }
1050             }
1051             {
1052                 \int_compare:nNnTF
1053                 { \tl_item:nn { ##1 } { 2 } }

```



```

1054 >
1055 { \g__postnotes_note_id_int }
1056 { \seq_put_right:Nn \l__postnotes_tmpb_seq { ##1 } }
1057 {
1058   \seq_put_right:Ne \l__postnotes_tmpa_seq
1059   { \tl_item:nn { ##1 } { 2 } }
1060 }
1061 }

```

Also no need for the F branch here, but for a different reason. The preprint label plays no role whatsoever if `couteraux=false`, so we just drop them altogether if found.

```

1062 }
1063 \seq_gset_eq:NN \g__postnotes_print_labelseq_queue_seq
1064 \l__postnotes_tmpa_seq
1065 \seq_gset_eq:NN \g__postnotes_labelseq_seq \l__postnotes_tmpb_seq
1066 }
1067 \group_end:
1068 }

```

(End of definition for __postnotes_split_labelseq:.)

`__postnotes_check_duplicates:N` provides a general procedure for handling cases of multiple passes of content. Ideally, the job should be done at `__postnotes_inhibit_note:F`, if at all possible. But, failing that, we can rely on the fact that the labels of `\postnotes` of measuring/trial passes, being delayed `\writes` (whatsits), don't end up being written to the `.aux` file. However, despite this being a general test, and a reasonable one, I'd like to restrain it's use to the minimum possible. Using this criterion across the board could result in large swings on the content of `\printpostnotes` and spurious warnings. Hence, we only apply this check for "eligible" items. For signaling this eligibility, the note must have been stored with the `\l__postnotes_maybe_multi_bool` set to `true`, which is then saved in the `multibool` property. One implication of this procedure is that, if there are any new notes marked as `multibool`, three rounds of compilation will be needed, since the labels of the printed notes will be written only on the second run and the document will thus require a third one to stabilize.

```

\__postnotes_check_duplicates:N \__postnotes_check_duplicates:N (\g__postnotes_print_queue_seq)
1069 \cs_new_protected:Npn \__postnotes_check_duplicates:N #1
1070 {

```

On a first run, don't even try to check for duplicates. Better *transitorily* let some duplicates pass than to drop every legitimate note.

```

1071 \bool_lazy_and:nnT
1072 { ! \g__postnotes_firstrun_bool }
1073 { ! \g__postnotes_couteraux_bool }
1074 {
1075   \group_begin:
1076   \seq_clear:N \l__postnotes_tmpa_seq
1077   \seq_map_inline:Nn #1
1078   {
1079     \bool_lazy_or:nnTF
1080     { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ mark @ ##1 } }

```

Always keep sections. Empty sections are discarded anyway, and they are unlikely to occur in places performing multiple passes.

```

1081         {
1082             \str_if_eq_p:ee
1083             { \__postnotes_prop_item:nn {##1} { type } } { section }
1084         }
1085         { \seq_put_right:Nn \l__postnotes_tmpa_seq {##1} }
1086     {

```

If `multibool` is true for the note, we drop it silently, otherwise we include it, but warn of possible duplicate.

```

1087         \str_if_eq:eeF
1088         { \__postnotes_prop_item:nn {##1} { multibool } }
1089         { true }
1090     {
1091         \seq_put_right:Nn \l__postnotes_tmpa_seq {##1}
1092         \bool_if:NT \l__postnotes_check_dupli_bool
1093         {
1094             \msg_warning:nne { postnotes } { possible-duplicate }
1095             { \__postnotes_prop_item:nn {##1} { mark } }
1096         }
1097     }
1098 }
1099 }
1100 \seq_gset_eq:NN #1 \l__postnotes_tmpa_seq
1101 \group_end:
1102 }
1103 }

```

(End of definition for __postnotes_check_duplicates:N.)

```

1104 \msg_new:nnn { postnotes } { possible-duplicate }
1105 { Possible~duplicate~*around*~note~'##1'~\msg_line_context:. }

```

```

\__postnotes_check_floats:N
    \__postnotes_check_floats:N <\g__postnotes_print_queue_seq>
1106 \cs_new_protected:Npn \__postnotes_check_floats:N #1
1107 {
1108     \bool_lazy_and:nnT
1109     { \l__postnotes_check_floats_bool }

```

Ditto. In this case, the queue is not touched, but it would still be a spurious warning.

```

1110     { ! \g__postnotes_firststrun_bool }
1111     {
1112         \group_begin:

```

Only compare sequence net of non-existing labels.

```

1113         \seq_set_filter:NNn \l__postnotes_tmpa_seq #1
1114         {
1115             \bool_lazy_or_p:nn
1116             { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ mark @ ##1 } }
1117             { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ section @ ##1 } }
1118         }

```

Not very expl3-y, I know. But I don't see a `\seq_if_eq:NNTF` available and, technically, sequences are just structured token lists.

```

1119         \tl_if_eq:NNF
1120         \g__postnotes_print_labelseq_queue_seq \l__postnotes_tmpa_seq
1121         { \msg_warning:nn { postnotes } { possible-shuffle } }
1122     \group_end:
1123 }
1124 }

```

(End of definition for `__postnotes_check_floats:N`.)

```

1125 \msg_new:nnn { postnotes } { possible-shuffle }
1126 { Possible-out-of-sequence-notes-due-to-floats~\msg_line_context:. }

```

`__postnotes_sort_queue:N` Sorting function for `__postnotes_print_notes:`.

```

        \__postnotes_sort_queue:N (\g__postnotes_print_queue_seq)
1127 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
1128 {
1129     \bool_if:NT \l__postnotes_sort_bool
1130     {
1131         \group_begin:
1132         \seq_gsort:Nn #1
1133         {
1134             \__postnotes_prop_get:nnN {##1} { pnsectid } \l__postnotes_tmpa_tl
1135             \__postnotes_prop_get:nnN {##2} { pnsectid } \l__postnotes_tmpb_tl
1136             \tl_if_eq:NNTF \l__postnotes_tmpa_tl \l__postnotes_tmpb_tl
1137             {
1138                 \__postnotes_prop_get:nnN {##1} { type } \l__postnotes_tmpa_tl
1139                 \__postnotes_prop_get:nnN {##2} { type } \l__postnotes_tmpb_tl
1140                 \bool_lazy_and:nnTF
1141                 { \str_if_eq_p:Vn \l__postnotes_tmpa_tl { note } }
1142                 { \str_if_eq_p:Vn \l__postnotes_tmpb_tl { note } }
1143                 {
1144                     \__postnotes_prop_get:nnN {##1}
1145                     { sortnum } \l__postnotes_tmpa_tl
1146                     \__postnotes_prop_get:nnN {##2}
1147                     { sortnum } \l__postnotes_tmpb_tl
1148                     \fp_compare:nNnTF
1149                     { \l__postnotes_tmpa_tl } > { \l__postnotes_tmpb_tl }
1150                     { \sort_return_swapped: }
1151                     { \sort_return_same: }
1152                 }
1153                 { \sort_return_same: }
1154             }
1155             { \sort_return_same: }
1156         }
1157     \group_end:
1158 }
1159 }

```

(End of definition for `__postnotes_sort_queue:N`.)

```

1160 \AddToHook { enddocument/afterlastpage }

```

```

1161 {
1162   \group_begin:
1163   \bool_if:NTF \g__postnotes_counteraux_bool
1164     {
1165       \seq_set_filter:NNn \l__postnotes_tmpa_seq \g__postnotes_labelseq_seq
1166       { \str_if_eq_p:ee { \tl_item:nn {#1} { 1 } } { mark } }
1167     }
1168     {
1169       \seq_set_filter:NNn \l__postnotes_tmpa_seq \g__postnotes_queue_seq
1170       { \str_if_eq_p:ee { \__postnotes_prop_item:nn {#1} { type } } { note } }
1171     }
1172   \seq_if_empty:NF \l__postnotes_tmpa_seq
1173     {
1174       \msg_warning:nne { postnotes } { stray-notes }
1175       { \seq_count:N \l__postnotes_tmpa_seq }
1176     }
1177   \group_end:
1178 }
1179 \msg_new:nnn { postnotes } { stray-notes }
1180 {
1181   There-are-~'#1'~stray~notes~after~the~last~'\iow_char:N\printpostnotes'~
1182   \msg_line_context:..~At~this~point,~they~are~lost.
1183 }

```

8 Headers

The headers infrastructure of postnotes is comprised of three basic parts:

1. For each `\postnote`, labels are set storing the page where the note occurs. Each note actually generates a pair of such labels, once when `\postnote` is called (with the mark), and another where the note is printed (in `\printpostnotes`). The former ones store `\thepage`, since we want the printed representation of it for typesetting purposes, the latter ones store the value of the page counter, since we don't need to typeset it, but do need to perform algebraic operations with it. These labels are set by `__postnotes_set_mark_page_label:nn`, `__postnotes_set_text_page_label:n`, and `__postnotes_set_print_page_label:n` at the appropriate places. The set of these labels provides a mapping from each note's "mark" and "text" to the page where it occurs.
2. This information set is processed by `__postnotes_get_headers_data:N` at the beginning of `__postnotes_print_notes:` to identify the first and last note of each page in `\printpostnotes`, and to generate a mapping from these first and last notes on each page to the pages where their corresponding marks occur. We also take the opportunity to enrich this mapping with other metadata of each note. So we get also mappings from the first and last note on each page to `\thechapter`, `\thesection`, and the name of the section in which they occur. These mappings are stored in property lists `\g__postnotes_header_{info}_first_prop` and `\g__postnotes_header_{info}_last_prop` where the key is the page in `\printpostnotes` where their note's content is typeset (or rather where it starts to be typeset, it is the page where the text's mark is printed).

- Based on these mappings, along the span of notes section we run `__postnotes_set_headers_vars_next`: at each `shipout/before` hook to set user facing variables for the *next* page, which will be available when their heading gets typeset. Given that at `shipout` we can rely on a correct value of the `page` counter, we use it as `key` to query the property lists generated in the previous step. These user facing variables are called `\pnhd<info>first` and `\pnhd<info>last`. Since we cannot rely on the `shipout` hook for the first page of `\printpostnotes`, `__postnotes_set_headers_vars_first`: is run at its beginning to ensure correct values are in place on the first page of the notes section.

These `\pnhd<info>first` and `\pnhd<info>last` variables can then be used to build simple functions which can be passed to mark commands to achieve rich contextual running headers.

<code>\pnhdpagefirst</code>	User facing variables, aimed at making available header data for the user. Setting these
<code>\pnhdpagelast</code>	variables with correct values at the moment the header gets typeset is <i>the</i> objective of
<code>\pnhdchapfirst</code>	the whole headers infrastructure of the package.
<code>\pnhdchapl原因</code>	
<code>\pnhdsectfirst</code>	1184 <code>\tl_new:N \pnhdpagefirst</code>
<code>\pnhdsectlast</code>	1185 <code>\tl_new:N \pnhdpagelast</code>
<code>\pnhdnamefirst</code>	1186 <code>\tl_new:N \pnhdchapfirst</code>
<code>\pnhdnamelast</code>	1187 <code>\tl_new:N \pnhdchapl原因</code>
	1188 <code>\tl_new:N \pnhdsectfirst</code>
	1189 <code>\tl_new:N \pnhdsectlast</code>
	1190 <code>\tl_new:N \pnhdnamefirst</code>
	1191 <code>\tl_new:N \pnhdnamelast</code>

(End of definition for `\pnhdpagefirst` and others.)

<code>\g__postnotes_header_page_first_prop</code>	Auxiliary variables for the headers infrastructure.
<code>\g__postnotes_header_page_last_prop</code>	1192 <code>\prop_new:N \g__postnotes_header_page_first_prop</code>
<code>\g__postnotes_header_chap_first_prop</code>	1193 <code>\prop_new:N \g__postnotes_header_page_last_prop</code>
<code>\g__postnotes_header_chap_last_prop</code>	1194 <code>\prop_new:N \g__postnotes_header_chap_first_prop</code>
<code>\g__postnotes_header_sect_first_prop</code>	1195 <code>\prop_new:N \g__postnotes_header_chap_last_prop</code>
<code>\g__postnotes_header_sect_last_prop</code>	1196 <code>\prop_new:N \g__postnotes_header_sect_first_prop</code>
<code>\g__postnotes_header_name_first_prop</code>	1197 <code>\prop_new:N \g__postnotes_header_sect_last_prop</code>
<code>\g__postnotes_header_name_last_prop</code>	1198 <code>\prop_new:N \g__postnotes_header_name_first_prop</code>
<code>\g__postnotes_header_prev_last_page_tl</code>	1199 <code>\prop_new:N \g__postnotes_header_name_last_prop</code>
<code>\g__postnotes_header_prev_last_chap_tl</code>	1200 <code>\tl_new:N \g__postnotes_header_prev_last_page_tl</code>
<code>\g__postnotes_header_prev_last_sect_tl</code>	1201 <code>\tl_new:N \g__postnotes_header_prev_last_chap_tl</code>
<code>\g__postnotes_header_prev_last_name_tl</code>	1202 <code>\tl_new:N \g__postnotes_header_prev_last_sect_tl</code>
<code>\l__postnotes_prev_text_page_tl</code>	1203 <code>\tl_new:N \g__postnotes_header_prev_last_name_tl</code>
<code>\l__postnotes_curr_text_page_tl</code>	1204 <code>\tl_new:N \l__postnotes_prev_text_page_tl</code>
<code>\l__postnotes_prev_mark_page_tl</code>	1205 <code>\tl_new:N \l__postnotes_curr_text_page_tl</code>
<code>\l__postnotes_prev_mark_chap_tl</code>	1206 <code>\tl_new:N \l__postnotes_prev_mark_page_tl</code>
<code>\l__postnotes_prev_mark_sect_tl</code>	1207 <code>\tl_new:N \l__postnotes_prev_mark_chap_tl</code>
<code>\l__postnotes_prev_mark_name_tl</code>	1208 <code>\tl_new:N \l__postnotes_prev_mark_sect_tl</code>
	1209 <code>\tl_new:N \l__postnotes_prev_mark_name_tl</code>

(End of definition for `\g__postnotes_header_page_first_prop` and others.)

`__postnotes_get_headers_data:N` Process header data for `__postnotes_set_headers_vars:n`.

`__postnotes_get_headers_data:N` (`\g__postnotes_print_queue_seq`)

```

1210 \cs_new_protected:Npn \__postnotes_get_headers_data:N #1
1211 {
1212   \group_begin:
1213     \tl_gclear:N \pnhdpagefirst
1214     \tl_gclear:N \pnhdpagelast
1215     \tl_gclear:N \pnhdchapfirst
1216     \tl_gclear:N \pnhdchaplast
1217     \tl_gclear:N \pnhdsectfirst
1218     \tl_gclear:N \pnhdsectlast
1219     \tl_gclear:N \pnhdnamefirst
1220     \tl_gclear:N \pnhdnamelast
1221     \prop_gclear:N \g__postnotes_header_page_first_prop
1222     \prop_gclear:N \g__postnotes_header_page_last_prop
1223     \prop_gclear:N \g__postnotes_header_chap_first_prop
1224     \prop_gclear:N \g__postnotes_header_chap_last_prop
1225     \prop_gclear:N \g__postnotes_header_sect_first_prop
1226     \prop_gclear:N \g__postnotes_header_sect_last_prop
1227     \prop_gclear:N \g__postnotes_header_name_first_prop
1228     \prop_gclear:N \g__postnotes_header_name_last_prop
1229     \tl_gclear:N \g__postnotes_header_prev_last_page_tl
1230     \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
1231     \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
1232     \tl_gclear:N \g__postnotes_header_prev_last_name_tl
1233     \tl_clear:N \l__postnotes_prev_text_page_tl
1234     \tl_clear:N \l__postnotes_curr_text_page_tl
1235     \tl_clear:N \l__postnotes_prev_mark_page_tl
1236     \tl_clear:N \l__postnotes_prev_mark_chap_tl
1237     \tl_clear:N \l__postnotes_prev_mark_sect_tl
1238     \tl_clear:N \l__postnotes_prev_mark_name_tl
1239     \seq_map_inline:Nn #1
1240     {
1241       \exp_args:Ne \tl_if_eq:nnT
1242       { \__postnotes_prop_item:nn {##1} { type } }
1243       { note }
1244       {
1245         \__postnotes_get_pageref:Nn
1246         \l__postnotes_curr_text_page_tl { text@ ##1 }
1247         \tl_if_empty:NF \l__postnotes_curr_text_page_tl
1248         {
1249           \tl_if_eq:NNTF
1250           \l__postnotes_prev_text_page_tl
1251           \l__postnotes_curr_text_page_tl
1252           {

```

We are on the same page as the previous note, just update the prev_mark data.

```

1253         \__postnotes_get_pageref:Nn
1254         \l__postnotes_prev_mark_page_tl { mark@ ##1 }
1255         \__postnotes_prop_get:nnN {##1} { thechapter }
1256         \l__postnotes_prev_mark_chap_tl
1257         \__postnotes_prop_get:nnN {##1} { thesection }
1258         \l__postnotes_prev_mark_sect_tl
1259         \__postnotes_prop_get:nnN {##1} { pnsectname }
1260         \l__postnotes_prev_mark_name_tl
1261       }

```

1262 {
 We are on the transition between two pages, current ID is the first note of the new page (or on the very first note of \printpostnotes, given \l__postnotes_prev_text_page_tl is initialized to empty).

Set 'last' values for previous page, based on the last valid prev_mark stored ones. There is no previous page to the first one of \printpostnotes, so we don't set 'last' values for it (conditioning on \l__postnotes_prev_text_page_tl being empty, which only occurs on the first note).

```
1263 \tl_if_empty:NF \l__postnotes_prev_text_page_tl
1264 {
1265   \prop_gput:Nee \g__postnotes_header_page_last_prop
1266   { \l__postnotes_prev_text_page_tl }
1267   { \l__postnotes_prev_mark_page_tl }
1268   \prop_gput:Nee \g__postnotes_header_chap_last_prop
1269   { \l__postnotes_prev_text_page_tl }
1270   { \l__postnotes_prev_mark_chap_tl }
1271   \prop_gput:Nee \g__postnotes_header_sect_last_prop
1272   { \l__postnotes_prev_text_page_tl }
1273   { \l__postnotes_prev_mark_sect_tl }
1274   \prop_gput:Nee \g__postnotes_header_name_last_prop
1275   { \l__postnotes_prev_text_page_tl }
1276   { \l__postnotes_prev_mark_name_tl }
1277 }
```

Set 'first' values for current page, based on the current note ID.

```
1278 \prop_gput:Nee \g__postnotes_header_page_first_prop
1279 { \l__postnotes_curr_text_page_tl }
1280 { \__postnotes_extract_pageref:n { mark@ ##1 } }
1281 \prop_gput:Nee \g__postnotes_header_chap_first_prop
1282 { \l__postnotes_curr_text_page_tl }
1283 { \__postnotes_prop_item:nn {##1} { thechapter } }
1284 \prop_gput:Nee \g__postnotes_header_sect_first_prop
1285 { \l__postnotes_curr_text_page_tl }
1286 { \__postnotes_prop_item:nn {##1} { thesection } }
1287 \prop_gput:Nee \g__postnotes_header_name_first_prop
1288 { \l__postnotes_curr_text_page_tl }
1289 { \__postnotes_prop_item:nn {##1} { pnsectname } }
```

Store prev_mark data for the first note on the page.

```
1290 \__postnotes_get_pageref:Nn
1291 \l__postnotes_prev_mark_page_tl { mark@ ##1 }
1292 \__postnotes_prop_get:nnN {##1} { thechapter }
1293 \l__postnotes_prev_mark_chap_tl
1294 \__postnotes_prop_get:nnN {##1} { thesection }
1295 \l__postnotes_prev_mark_sect_tl
1296 \__postnotes_prop_get:nnN {##1} { pnsectname }
1297 \l__postnotes_prev_mark_name_tl
```

Set \l__postnotes_prev_text_page_tl for the next page (\l__postnotes_curr_text_page_tl is never empty at this point, since we conditioned to it).

```
1298 \tl_set:NV \l__postnotes_prev_text_page_tl
1299 \l__postnotes_curr_text_page_tl
1300 }
1301 }
```

```

1302     }
1303 }

```

We can't catch the transition from the last page of `\printpostnotes` to the following one through the mapping above, but the `prev_mark` values of the last note in the loop are the ones we want, so we set 'last' values for the last page based on them.

```

1304 \tl_if_empty:NF \l__postnotes_prev_text_page_tl
1305 {
1306   \prop_gput:Nee \g__postnotes_header_page_last_prop
1307   { \l__postnotes_prev_text_page_tl }
1308   { \l__postnotes_prev_mark_page_tl }
1309   \prop_gput:Nee \g__postnotes_header_chap_last_prop
1310   { \l__postnotes_prev_text_page_tl }
1311   { \l__postnotes_prev_mark_chap_tl }
1312   \prop_gput:Nee \g__postnotes_header_sect_last_prop
1313   { \l__postnotes_prev_text_page_tl }
1314   { \l__postnotes_prev_mark_sect_tl }
1315   \prop_gput:Nee \g__postnotes_header_name_last_prop
1316   { \l__postnotes_prev_text_page_tl }
1317   { \l__postnotes_prev_mark_name_tl }
1318 }
1319 \group_end:
1320 }

```

(End of definition for `__postnotes_get_headers_data:N`.)

The sequence of pages processed in `__postnotes_get_headers_data:N` is not ensured to be continuous, since not every page of `\printpostnotes` starts a note. There may be notes that fill whole pages, or the last page of the notes may end with a note that started on the penultimate page. We must handle this case at `__postnotes_set_headers_vars:n`. For every page for which there is information provided by `__postnotes_get_headers_data:N` we store a `header_prev_last` (the last value of the previous header) for each of the variables of interest. If the next page is skipped in the sequence (no notes starting on it), we can use these stored values to set both 'first' and 'last' variables based on them for that page.

`__postnotes_set_headers_vars:n` Set user facing variables based on data generated by `__postnotes_get_headers_data:N`.

```

\__postnotes_set_headers_vars:n {<page number>}

1321 \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
1322 {
1323   \group_begin:
1324   \prop_get:NnNTF \g__postnotes_header_page_first_prop
1325   {#1} \l__postnotes_tmpa_tl
1326   { \tl_gset:NV \pnhdpagefirst \l__postnotes_tmpa_tl }
1327   { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
1328   \prop_get:NnNTF \g__postnotes_header_page_last_prop
1329   {#1} \l__postnotes_tmpa_tl
1330   {
1331     \tl_gset:NV \pnhdpagelast \l__postnotes_tmpa_tl
1332     \tl_gset:NV \g__postnotes_header_prev_last_page_tl
1333     \l__postnotes_tmpa_tl
1334   }

```



```

1335     { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
1336 \prop_get:NnNTF \g__postnotes_header_chap_first_prop
1337   {#1} \l__postnotes_tmpa_tl
1338   { \tl_gset:NV \pnhdchapfirst \l__postnotes_tmpa_tl }
1339   { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
1340 \prop_get:NnNTF \g__postnotes_header_chap_last_prop
1341   {#1} \l__postnotes_tmpa_tl
1342   {
1343     \tl_gset:NV \pnhdchaplast \l__postnotes_tmpa_tl
1344     \tl_gset:NV \g__postnotes_header_prev_last_chap_tl
1345     \l__postnotes_tmpa_tl
1346   }
1347   { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
1348 \prop_get:NnNTF \g__postnotes_header_sect_first_prop
1349   {#1} \l__postnotes_tmpa_tl
1350   { \tl_gset:NV \pnhdsectfirst \l__postnotes_tmpa_tl }
1351   { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
1352 \prop_get:NnNTF \g__postnotes_header_sect_last_prop
1353   {#1} \l__postnotes_tmpa_tl
1354   {
1355     \tl_gset:NV \pnhdsectlast \l__postnotes_tmpa_tl
1356     \tl_gset:NV \g__postnotes_header_prev_last_sect_tl
1357     \l__postnotes_tmpa_tl
1358   }
1359   { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
1360 \prop_get:NnNTF \g__postnotes_header_name_first_prop
1361   {#1} \l__postnotes_tmpa_tl
1362   { \tl_gset:NV \pnhdnamefirst \l__postnotes_tmpa_tl }
1363   { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
1364 \prop_get:NnNTF \g__postnotes_header_name_last_prop
1365   {#1} \l__postnotes_tmpa_tl
1366   {
1367     \tl_gset:NV \pnhdnamelast \l__postnotes_tmpa_tl
1368     \tl_gset:NV \g__postnotes_header_prev_last_name_tl
1369     \l__postnotes_tmpa_tl
1370   }
1371   { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
1372 \group_end:
1373 }
1374 \cs_generate_variant:Nn \__postnotes_set_headers_vars:n { e }

```

(End of definition for __postnotes_set_headers_vars:n.)

__postnotes_set_headers_vars_next: The functions that actually call __postnotes_set_headers_vars:n at the appropriate contexts with appropriate page values. Though we set __postnotes_set_headers_vars_next: to run at every shipout/before hook of the document, it is made no-op by \g__postnotes_header_vars_next_bool which only has a true value inside \printpostnotes. __postnotes_set_headers_vars_first: must set a label and retrieve its value to be able to have a reliable value of its own page.

```

1375 \AddToHook { shipout/before } [ ./header ]
1376   { \__postnotes_set_headers_vars_next: }
1377 \bool_new:N \g__postnotes_header_vars_next_bool
1378 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:

```

```

1379 {
1380   \bool_if:NT \g__postnotes_header_vars_next_bool
1381     { \__postnotes_set_headers_vars:e { \int_eval:n { \c@page + 1 } } }
1382 }
1383 \cs_new_protected:Npn \__postnotes_set_headers_vars_first:
1384 {
1385   \__postnotes_set_print_page_label:e
1386     { \int_use:N \g__postnotes_print_postnotes_int }
1387   \__postnotes_set_headers_vars:e
1388     {
1389       \__postnotes_extract_pageref:e
1390       { print@ \int_use:N \g__postnotes_print_postnotes_int }
1391     }
1392 }

```

(End of definition for `__postnotes_set_headers_vars_next:` and `__postnotes_set_headers_vars_first:`.)

`\pnheaderdefault` A basic header function to be used as default in the `heading` option. It produces a header in the form “Notes to pages N–M”, with a text which can be localized (see Section 10).

`\pnheaderdefault`

```

1393 \NewDocumentCommand \pnheaderdefault {}
1394 {
1395   \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
1396     { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
1397     { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
1398 }

```

(End of definition for `\pnheaderdefault`.)

9 Compatibility

A dedicated temp variable for restoring data.

```

1399 \tl_new:N \l__postnotes_restore_tmp_tl

```

`\caption`

For `\caption`’s possible two passes. This catches more than just captions, of course, but is not overkill. A hook to `\@makecaption` would be better, but `ltxcmds` does not allow it, and using lower level methods for this is a bad idea.

From the user’s perspective, one-line captions will just work. For two-line captions, there are two alternatives: i) `\stepcounter{postnote}` before the caption, then call `\postnote` with `mark=\arabic{postnote}`; or ii) right before the caption, call `\postnote[nomark]{\label{mynote}...}`, then use `\postnoteref{mynote}` inside the caption.

```

1400 \AddToHook { postnotes/note/begin } [ ./compat/caption ]
1401 { \cs_if_exist:NT \@capttype { \postnotesetup { maybemulti } } }

```

biblatex

Thanks Moritz Wemheuer: https://tex.stackexchange.com/q/597359#comment1594585_597389.

```
1402 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1403 {
```

Let biblatex know we are in a “notes” context. See <https://tex.stackexchange.com/a/304464>, including comments.

```
1404     \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1405     { \toggletrue { blx@footnote } }
```

Make biblatex’s `\mkbibendnote` use `\postnote`. This is very likely desired in most cases, but may occasionally not be, so we add it to an individually labeled hook, which can be disabled with `\RemoveFromHook{begindocument/before}[postnotes/mkbibendnote]` in the preamble.

```
1406     \AddToHook { begindocument/before } [ postnotes/compat/biblatex ]
1407     {
1408         \cs_set:Npn \blx@theendnote { \postnote }
1409         \cs_set:Npn \blx@theendnotetext { \blx@err@endnote \footnotetext }
1410     }
1411 }
1412 <*gobble>
```

I had made an initial experimental attempt to support biblatex’s `refsegments`, `refcontexts` and `refsections`. However, this attempt was rash. Even if I could get many example files to work for `refsegments` and `refcontexts`, I could not do so for `refsections`. More importantly, with this partial implementation, I could also generate documents which confused biblatex more than it helped. Things I couldn’t understand well, or fix. All in all, I don’t think this partial implementation is tenable, and I could not take it further. Hence, `postnotes` support for this feature set of biblatex will depend, as it should, on proper upstream support for “saving” and “restoring” citation “context” information.

I have made a feature request at biblatex for this (<https://github.com/plk/biblatex/issues/1226>), which was (understandably) classified as “long term, no promises”.

The attempt was the following (currently “gobbled” from the package):

```
1413 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1414 {
```

Store biblatex variables for each note.

```
1415     \AddToHook { postnotes/note/store } [ postnotes/compat/biblatex ]
1416     {
1417         \prop_gput:cne { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1418         { biblatex@refsection } { \int_use:N \c@refsection }
1419         \prop_gput:cne { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1420         { biblatex@refsegment } { \int_use:N \c@refsegment }
1421         \prop_gput:cne { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1422         { biblatex@refcontextbool }
1423         { \iftoggle { blx@refcontext } { true } { false } }
1424         \prop_gput:cnV { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1425         { biblatex@refcontext } \blx@refcontext@context
1426     }
```

biblatex setup, once for `\printpostnotes` call.

```

1427 \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1428 {
1429   \__postnotes_biblatex_endrefcontext_local:
1430   \__postnotes_biblatex_citereset_local:
1431 }

```

Restore biblatex variables for each note.

```

1432 \AddToHook { postnotes/print/note/begin } [ postnotes/compat/biblatex ]
1433 {
1434   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1435   { biblatex@refsection } \l__postnotes_restore_tmp_tl
1436   \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
1437   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1438   { biblatex@refsegment } \l__postnotes_restore_tmp_tl
1439   \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
1440   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1441   { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
1442   \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
1443   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1444   { biblatex@refcontext } \l__postnotes_restore_tmp_tl
1445   \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
1446 }

```

Auxiliary functions.

`__postnotes_biblatex_endrefcontext_local:` Replicate the job of `\endrefcontext`, but with local effects, restrained to the group of `\printpostnotes`.

```

1447 \cs_new_protected:Npn \__postnotes_biblatex_endrefcontext_local:
1448 {
1449   \togglefalse { blx@refcontext }
1450   \tl_clear:N \blx@refcontext@labelprefix
1451   \tl_clear:N \blx@refcontext@labelprefix@real
1452   \tl_set:Ne \blx@refcontext@sortingtemplatename { \blx@sorting }
1453   \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
1454   \tl_set:Nn \blx@refcontext@uniquenametemplatename { global }
1455   \tl_set:Nn \blx@refcontext@labelalphanametemplatename { global }
1456   \blx@edef@refcontext
1457   {
1458     \blx@refcontext@sortingtemplatename /
1459     \blx@refcontext@sortingnamekeytemplatename /
1460     /
1461     \blx@refcontext@uniquenametemplatename /
1462     \blx@refcontext@labelalphanametemplatename
1463   }
1464 }

```

(End of definition for `__postnotes_biblatex_endrefcontext_local:.`)

`__postnotes_biblatex_citereset_local:` Replicate the job of `\citereset`, but with local effects, restrained to the group of `\printpostnotes`.

```

1465 \cs_new_protected:Npn \__postnotes_biblatex_citereset_local:
1466 {

```

```

\global\cslet{blx@bsee@the\c@refsection}\@empty
\global\cslet{blx@fsee@the\c@refsection}\@empty
1467     \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1468     \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }

\blx@ibidreset@force
1469     \undef \blx@lastkey@text
1470     \undef \blx@lastkey@foot

\blx@idemreset@force
1471     \undef \blx@lasthash@text
1472     \undef \blx@lasthash@foot

\blx@opcitreset@force
1473     \clist_map_inline:Nn \blx@trackhash@text
1474     { \csundef { blx@lastkey@text@ ##1 } }
1475     \tl_clear:N \blx@trackhash@text
1476     \clist_map_inline:Nn \blx@trackhash@foot
1477     { \csundef { blx@lastkey@foot@ ##1 } }
1478     \tl_clear:N \blx@trackhash@foot

\blx@loccitreset@force
1479     \clist_map_inline:Nn \blx@trackkeys@text
1480     { \csundef { blx@lastnote@text@ ##1 } }
1481     \tl_clear:N \blx@trackkeys@text
1482     \clist_map_inline:Nn \blx@trackkeys@foot
1483     { \csundef { blx@lastnote@foot@ ##1 } }
1484     \tl_clear:N \blx@trackkeys@foot

and all of them do:
1485     \cs_set_eq:NN \blx@lastmpfn \z@
1486     }

(End of definition for \__postnotes_biblatex_citereset_local:.)

1487 }

biblatex's refsections, contrary to refsegments and refcontexts which are handled in the LATEX side of things (as far as I can tell), need to go through biber, and must have correct corresponding citation data written to the .bcf file. And the way \refsection is implemented presumes each section is only ever begun once (fair...), thus making it difficult to “reopen” it, or append new citations to it later on, when the notes are printed. The start of a refsection must be registered on the .bcf file, and this is done by \refsection (and its auxiliary functions). However, a number of its characteristics make things particularly difficult for the purpose at hand: i) it unconditionally sets a label for the section which, of course, cannot be done twice; and, critically, ii) the optional argument of the environment (which receives the <resources>) is used to set a local assignment to \blx@bibfiles, based on which the relevant information is written to the .bcf file, and when the group closes the information is gone. My best attempt is below but it is not good. It feels a wrong approach to “go around” the intended use of \refsection so much, and it can't handle at all its optional argument, for the reasons above. It's also incomplete, since it does not handle restoring \l__postnotes_biblatex_orig_refsection_tl.

1488 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1489 {

```

```

1490 \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1491 \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1492 \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1493 {
1494   \tl_set:Ne \l__postnotes_biblatex_orig_refsection_tl
1495     { \int_use:N \c@refsection }
1496   \tl_gset:Ne \g__postnotes_biblatex_prev_refsection_tl
1497     { \l__postnotes_biblatex_orig_refsection_tl }
1498 }
1499 \AddToHook { postnotes/print/note/begin } [ postnotes/compat/biblatex ]
1500 {
1501   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1502     { biblatex@refsection } \l__postnotes_restore_tmp_tl
1503   \tl_if_eq:NNF
1504     \l__postnotes_restore_tmp_tl
1505     \g__postnotes_biblatex_prev_refsection_tl
1506     {
1507       \int_set:Nn \c@blx@maxsection
1508         { \l__postnotes_restore_tmp_tl - 1 }
1509       \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1510         \l__postnotes_restore_tmp_tl
1511       \group_begin:
1512         \cs_set_eq:NN \label \use_none:n
1513         \cs_set_eq:NN \blx@info \use_none:n
1514         \blx@endrefsection
1515         \refsection
1516       \group_end:
1517     }
1518 }
1519 }
1520 </gobble>

```

zref-user

`\l__postnotes_note_zlabel_str` Even though the `zlabel` option is provided only when `zref-user` is loaded, `\l__postnotes_note_zlabel_str` must be unconditionally defined, since it is presumed to exist by `__postnotes_set_user_labels:` and elsewhere.

```

1521 \str_new:N \l__postnotes_note_zlabel_str

```

(End of definition for `\l__postnotes_note_zlabel_str`.)

```

1522 \AddToHook { package/zref-user/after } [ ./compat/zref-user ]
1523 {

```

Provide `zlabel` option.

```

1524   \keys_define:nn { postnotes/note }
1525     {
1526       zlabel .str_set:N = \l__postnotes_note_zlabel_str ,
1527       zlabel .value_required:n = true ,
1528     }

```

Provide property to store the mark for measuring passes.

```

1529   \zref@newprop { postnote@mark } [] { \l__postnotes_mark_tl }
1530   \AddToHook { postnotes/note/begin } [ postnotes/compat/zref-user ]
1531     { \zref@localaddprop { main } { postnote@mark } }

```

`\postnotezref` Provide `\postnotezref`.

```
\postnotezref(*){<label>}
```

```
1532 \NewDocumentCommand \postnotezref { s m }  
1533 { \__postnotes_note_zref:nn {#1} {#2} }
```

(End of definition for `\postnotezref`.)

`__postnotes_note_zref:nn` The internal version of `\postnotezref`.

```
\__postnotes_note_zref:nn {<star bool>} {<label>}
```

```
1534 \tl_new:N \l__postnotes_note_zref_zlabel_tl  
1535 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2  
1536 {  
1537   \group_begin:  
1538   \tl_set:Nn \l__postnotes_note_zref_zlabel_tl {#2}  
1539   \__postnotes_typeset_mark_wrapper:nnn  
1540   {  
1541     \bool_lazy_all:nTF  
1542     {  
1543       { ! #1 }  
1544       { \l__postnotes_hyperlink_bool }  
1545       { \l__postnotes_zrefhyperref_bool }  
1546     }  
1547     {  
1548       \hyperlink  
1549       { \zref@extractdefault {#2} { anchor } { } }  
1550       { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }  
1551     }  
1552     { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }  
1553   }  
1554   { \tag_socket_use:n { postnotes/postnotezref/begin } }  
1555   { \tag_socket_use:n { postnotes/postnotezref/end } }  
1556   \group_end:  
1557 }
```

(End of definition for `__postnotes_note_zref:nn`.)

```
1558 }  
1559 \bool_new:N \l__postnotes_zrefhyperref_bool  
1560 \AddToHook { package/zref-hyperref/after } [ ./compat/zref-hyperref ]  
1561 { \bool_set_true:N \l__postnotes_zrefhyperref_bool }
```

zref-clever

```
1562 \AddToHook { package/zref-clever/after } [ ./compat/zref-clever ]  
1563 {  
1564   \zcsetup  
1565   {  
1566     countertype = { postnote = endnote } ,  
1567     countertype = { postnotetext = endnote } ,  
1568   }
```

```

1569 \AddToHook { postnotes/print/begin } [ postnotes/compat/zref-clever ]
1570 { \zcsetup { counterresetby = { postnotetext = postnotesection } } }
1571 }

```

zref-check

```

1572 \AddToHook { package/zref-check/after } [ ./compat/zref-check ]
1573 {
1574 \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1575 {
1576 \AddToHook { postnotes/note/store } [ postnotes/compat/zref-check ]
1577 {
1578 \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1579 { zref-check@abschap } { \int_use:N \c@zc@abschap }
1580 \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1581 { zref-check@abssec } { \int_use:N \c@zc@abssec }
1582 }
1583 \AddToHook { postnotes/print/note/begin } [ postnotes/compat/zref-check ]
1584 {
1585 \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1586 { zref-check@abschap } \l__postnotes_restore_tmp_tl
1587 \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1588 \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1589 { zref-check@abssec } \l__postnotes_restore_tmp_tl
1590 \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1591 }
1592 }
1593 { }
1594 }

```

amsmath

```

1595 \AddToHook { package/amsmath/after } [ ./compat/amsmath ]
1596 {

```

Testing for `\ifmeasuring@` is sufficient to get things right for the measuring passes in math environments.

```

1597 \AddToHook { postnotes/note/inhibit } [ postnotes/compat/amsmath ]
1598 {
1599 \legacy_if:nT { measuring@ }
1600 {
1601 \bool_set_true:N \l__postnotes_inhibit_note_bool
1602 \bool_set_true:N \l__postnotes_print_plain_mark_bool
1603 \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1604 }
1605 }

```

However, the `\text` macro, defined by `amstext` (required by `amsmath`), poses problems if its own. Despite my best efforts, I could not salvage things from the use of `\mathchoice` and the redefinitions of `\setcounter` and `\addtocounter` performed by `amstext`. Setting `maybemulti` when `firstchoice@` is false grants us a working situation for display style. But the use of `\postnote` inside `\text` (and, if `amsmath` is loaded, `\textnormal`, `\textup`, etc.) in inline math environments is not supported. If a note really needs to be there, one can use the `nomark` option and `\postnoteref`. Things should work in text mode and in display style. For some related discussion with regard to footnotes, see

<https://tex.stackexchange.com/a/82820> and, in particular, Barbara Beeton's comment: "This is certainly bravura code. I do hope it doesn't result in a request to add `\footnote` capabilities to `amsmath`'s multi-line display facilities. (The answer will almost certainly be no. We agree with Kopka & Daly.)"

```
1606 \AddToHook { postnotes/note/begin } [ postnotes/compat/amsmath ]
1607     { \legacy_if:nF { firstchoice@ } { \postnotessetup { maybemulti } } }
1608 }
```

csquotes

```
1609 \AddToHook { package/csquotes/after } [ ./compat/csquotes ]
1610 {
1611   \bool_new:N \l__postnotes_csquotes_measuring_bool
1612   \BlockquoteDisable
1613   { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1614   \AddToHook { postnotes/note/inhibit } [ postnotes/compat/csquotes ]
1615   {
1616     \bool_if:NT \l__postnotes_csquotes_measuring_bool
1617     {
1618       \bool_set_true:N \l__postnotes_inhibit_note_bool
1619       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1620       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1621     }
1622   }
1623 }
```

tabularx

For the identification of the trial passes in `tabularx`, see <https://tex.stackexchange.com/a/640035> (including discussion in the comments, thanks David Carlisle), and also <https://tex.stackexchange.com/a/227155> and <https://tex.stackexchange.com/a/352134>.

```
1624 \AddToHook { package/tabularx/after } [ ./compat/tabularx ]
1625 {
1626   \bool_new:N \l__postnotes_tabularx_inside_env_bool
1627   \AddToHook { env/tabularx/begin } [ postnotes/compat/tabularx ]
1628   {
1629     \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1630     \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1631   }
1632   \AddToHook { postnotes/note/inhibit } [ postnotes/compat/tabularx ]
1633   {
1634     \bool_lazy_and:nnT
1635     { \l__postnotes_tabularx_inside_env_bool }
1636     { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1637     {
1638       \bool_set_true:N \l__postnotes_inhibit_note_bool
1639       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1640       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1641     }
1642   }
1643   \AddToHook { package/xltabular/after } [ postnotes/compat/xltabular ]
1644   {
1645     \AddToHook { env/xltabular/begin } [ postnotes/compat/xltabular ]
1646     {
```

```

1647         \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1648         \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1649     }
1650 }
1651 }

```

tabularray

```

1652 \AddToHook { package/tabularray/after } [ ./compat/tabularray ]
1653 {

```

Since version 2023A, from 2023-03-01, `tabularray` offers the `\lTblrMeasuringBool` which is true when measuring and false otherwise. See <https://tex.stackexchange.com/q/675818> and <https://github.com/lvjr/tabularray/issues/179> (thanks Ulrike Fischer).

```

1654     \bool_if_exist:NT \lTblrMeasuringBool
1655     {

```

I’d be inclined to restrict the inhibition effect to known `tabularray` environments to “keep things under control”. However this is a dedicated and public boolean, and users can create arbitrary new `tabularray` environments with `\NewTblrEnviron`, which we either wouldn’t catch or have to provide an user interface for. So, for the time being, let’s trust this boolean won’t be misused by third-parties or users. Note that setting `\l__postnotes_print_plain_mark_stepcounter_bool` to true presumes `tabularray`’s `counter` module is enabled. But, since this is the only way to get the measuring right in this context if there is more than one `\postnote` inside a given table, `postnotes` expects and requires the `counter` module.

```

1656     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/tabularray ]
1657     {
1658         \bool_if:NT \lTblrMeasuringBool
1659         {
1660             \bool_set_true:N \l__postnotes_inhibit_note_bool
1661             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1662             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1663         }
1664     }
1665 }
1666 }

```

PDF Tagging (experimental)

Note: All of this mostly presumes `\DocumentMetadata{testphase=phase-III}` and was tested with it. For `listenv=none`, I’d expect things to work with `phase-II`, but this is only lightly tested.

A first thing to consider in tagging endnotes is how we want to represent them in the PDF structure. My first thought, for lack of another, was: emulate footnotes. There’s no relevant semantic difference at the structure level between the two, and the tagging support for footnotes was done by the pros. And one distinctive characteristic the the footnotes tagging is that the footnote itself is placed in the structure as a child to the (parent) `text` element which surrounds the footnote mark. However, for endnotes this introduces a number of problems and complicates things. While footnotes “float around” and have no real structure of their own, that is not true for endnotes. Endnotes comprise a proper document section, and may be printed in a list environment, etc. So when the tagging of footnotes places the footnote structure element as a child element of the

mark’s surrounding text, this is arguably for a lack of other options. Where else, after all? Indeed, a typical `html` would render footnotes at the end of the page, and not inline. True the normal `html` page is much smaller than our typical PDF, but the point stands. We can have a hover over call out for footnotes, but the same could be done for end notes, regardless of the parent-child relation (as long as the required cross-references are in place). On the other hand, for endnotes this is not an issue: they have a natural place to be plugged into. Furthermore, making an endnote a child of the text surrounding its mark leaves an empty “skeleton” of the endnotes section: the heading, the list structure, etc. Technically, we could clean that too, but clearly that’s not the way to go. . . .

Finally, the parent-child relation is not required by PDF standards for the relevant structure types. The PDF 2.0 standard (ISO 32000-2:2020), says the following about `FENote`:

Used to markup footnotes and endnotes. Footnotes and endnotes are content that is not normally read as part of the enclosing content from which it is referenced, but rather consulted at the reading person’s discretion. In order for text to be considered a footnote or endnote, there should be a reference from the enclosing content to the footnote or endnote. Such reference may be achieved by means of a Link structure element through a structure destination in its link annotation (see “Table 368 — General inline level structure types”), or use of Ref in structure elements (see “Table 355 — Entries in a structure element dictionary”).

The PDF 1.7 standard (PDF 32000-1:2008), says the following about `Note`:

An item of explanatory text, such as a footnote or an endnote, that is referred to from within the body of the document. It may have a label (structure type `Lbl`; see “List Elements” in 14.8.4.3, “Block-Level Structure Elements”) as a child. The note may be included as a child of the structure element in the body text that refers to it, or it may be included elsewhere (such as in an endnotes section) and accessed by means of a reference (structure type `Reference`).

Tagged PDF does not prescribe the placement of footnotes in the page content order. They may be either inline or at the end of the page, at the discretion of the conforming writer.

So, the note *may* be included as a child of the surrounding text, but that’s not required (PDF 2.0 does not even mention that). What is required is the reference between the elements. All in all, let’s not follow footnotes in establishing the parent-child relation.

Another smaller but related issue is how to treat the list structure in `\printpostnotes` when `listenv` is used. The natural thing here would be to use an `enumerate` type list (or, in PDF lingo, for the `ListNumbering` attribute to be `Ordered`), where the mark is used as the item’s label (even if, technically, we use the list like a `description` in that we feed the label of every `\item` and though there is an implicit underlying counter, the list itself has no bearing upon it). The problem here is that the PDF 2.0 standards determine that the `FENote` structure element cannot be a child of a `LI` (list item). However, at least in principle, we also would like to have the `endnotelabel` element to be a child of the `endnote` element. Thus, we have a conflict, the mc-chunk can only be used once, and can be either the `Lbl` of the `LI`, or the `endnotelabel` for the `endnote`. Currently, for the list case, I’m using an `EndnotesList` class, which we define to have `ListNumbering`

as `Ordered`, with the mark as `Lbl` for `LI`, and letting `endnote` be a child of `LBody`. In a way, the possibility of exporting the tagged content to different formats makes me think that this is the most appropriate for the this case. For the case of `listenv=none`, the `endnotes` were made child of the `Sect` in which they occur. The `endnotelabel` was then included as part (child) of the `endnote`. In this, this treatment emulates the one given for footnotes in the kernel. But, at the same time, it is less than ideal for machine readability purposes, since whether the `endnotelabel` is part of `endnote` or not depends on if there is a list environment involved. Alas, I see no easy way around the PDF standard restriction for the list case.

On the \LaTeX side of things, adding support for tagging entails two basic tasks: i) applying the tagging markup at the appropriate places; ii) generating references between the “mark(s)” and the “text” (plus cross-reference commands to their respective targets).

Regarding tagging references, we have three different cases: i) a regular `\postnote`; ii) a `\postnoteref` to a note labeled from inside the note; and iii) a `\postnoteref` to a note labeled with the `label` option.

For regular `\postnotes` it is trivial to establish the reference using the `label / ref` options of `tagpdf`.

For `\postnoterefs`, however labeled, the connection *must* be established at `\postnoteref`, for the simple fact that any `\postnote` can be referenced by arbitrarily many `\postnoterefs`. So we need to be able to retrieve the note ID from the “text” to which the reference refers to at `\postnoteref`. However, at `\postnoteref` the only information we have is the `\langle label \rangle` but, since it is unique, we can establish a connection through it.

When the label is set from inside the note, it is actually set at `\printpostnotes`, at which place we have access to `\l_postnotes_print_note_id_tl` so we can use the `\langle label \rangle` to pass that information around to `\postnoteref`. With a standard `\label` we must set an additional `lproperties` label, using the new `label` hook, which `\postnoteref` can retrieve from its own `\langle label \rangle` argument. For `\zlabel` this particular task is trivial, since we can simply add a property to store the note ID with the label, which `\postnotezref` can extract.

When the label is set from the option, things are slightly more complicated, because the label is set at `\postnote`, at which place we do not have access to the note ID of the “text”. What we do here is then store the label with the note and restore it later at `\printpostnotes` as usual. Then, at that point, we can set an auxiliary label which can be retrieved from `\postnoteref` from its own `\langle label \rangle` argument, as we did for the case of labels inside the note. Auxiliary labels are handled with `lproperties` labels.

Unconditionally define tagging support sockets.

```

1667 \socket_new:nn { tagssupport/postnotes/mark/begin }{ 0 }
1668 \socket_new:nn { tagssupport/postnotes/mark/end }{ 0 }
1669 \socket_new:nn { tagssupport/postnotes/nomark/begin }{ 0 }
1670 \socket_new:nn { tagssupport/postnotes/nomark/end }{ 0 }
1671 \socket_new:nn { tagssupport/postnotes/multisep/begin }{ 0 }
1672 \socket_new:nn { tagssupport/postnotes/multisep/end }{ 0 }
1673 \socket_new:nn { tagssupport/postnotes/printlist/begin }{ 0 }
1674 \socket_new:nn { tagssupport/postnotes/printlist/end }{ 0 }
1675 \socket_new:nn { tagssupport/postnotes/printnote/begin }{ 0 }
1676 \socket_new:nn { tagssupport/postnotes/printnote/end }{ 0 }
1677 \socket_new:nn { tagssupport/postnotes/printmark/begin }{ 0 }
1678 \socket_new:nn { tagssupport/postnotes/printmark/end }{ 0 }
1679 \socket_new:nn { tagssupport/postnotes/printtext/begin }{ 0 }

```

```

1680 \socket_new:nn { tagsupport/postnotes/printtext/end }{ 0 }
1681 \socket_new:nn { tagsupport/postnotes/postnoteref/begin }{ 0 }
1682 \socket_new:nn { tagsupport/postnotes/postnoteref/end }{ 0 }
1683 \socket_new:nn { tagsupport/postnotes/postnoteref/begin }{ 0 }
1684 \socket_new:nn { tagsupport/postnotes/postnoteref/end }{ 0 }

1685 \bool_lazy_and:nnT
1686   { \cs_if_exist_p:N \tag_if_active_p: }
1687   { \tag_if_active_p: }
1688   {

```

FIXME Review or remove these settings if/when they are included upstream (see <https://github.com/latex3/tagging-project/issues/728>).

```

1689   \tagpdfsetup
1690   {
1691     role/new-tag = { tag=endnote, role=FENote } ,
1692     role/new-tag = { tag=endnotemark, role=Lbl } ,
1693     role/new-tag = { tag=endnotelabel, role=Lbl } ,
1694     role/new-attribute =
1695       { EndnoteType } { /O /FENote /NoteType /Endnote } ,
1696     role/new-attribute =
1697       { EndnotesList } { /O /List /ListNumbering /Ordered } ,
1698   }

\postnote
1699   \socket_new_plug:nnn { tagsupport/postnotes/mark/begin } { default }
1700   {
1701     \tag_mc_end_push:
1702     \tag_struct_begin:n
1703     {
1704       tag = endnotemark ,
1705       label = { postnotemark. \l_postnotes_note_id_tl } ,
1706       ref = { postnote. \l_postnotes_note_id_tl } ,
1707     }
1708     \__postnotes_tagsup_store_sstructnum:nN
1709     { postnotemark } \l_postnotes_note_id_tl
1710     \tag_mc_begin:n { }
1711   }
1712   \socket_new_plug:nnn { tagsupport/postnotes/mark/end } { default }
1713   {
1714     \tag_mc_end:
1715     \tag_struct_end: % endnotemark
1716     \tag_mc_begin_pop:n { }
1717   }
1718   \socket_new_plug:nnn { tagsupport/postnotes/nomark/begin } { default }
1719   {
1720     \tag_struct_begin:n
1721     {
1722       tag = NonStruct ,
1723       label = { postnotemark. \l_postnotes_note_id_tl } ,
1724       ref = { postnote. \l_postnotes_note_id_tl } ,
1725     }
1726     \__postnotes_tagsup_store_sstructnum:nN
1727     { postnotemark } \l_postnotes_note_id_tl
1728   }

```

```

1729 \socket_new_plug:nmn { tagsupport/postnotes/nomark/end } { default }
1730 { \tag_struct_end: } % NonStruct
1731 \socket_assign_plug:nm { tagsupport/postnotes/mark/begin } { default }
1732 \socket_assign_plug:nm { tagsupport/postnotes/mark/end } { default }
1733 \socket_assign_plug:nm { tagsupport/postnotes/nomark/begin } { default }
1734 \socket_assign_plug:nm { tagsupport/postnotes/nomark/end } { default }
multiple
1735 \socket_new_plug:nmn { tagsupport/postnotes/multisep/begin } { default }
1736 {
1737   \tag_mc_end_push:
1738   \tag_mc_begin:n { artifact }
1739 }
1740 \socket_new_plug:nmn { tagsupport/postnotes/multisep/end } { default }
1741 {
1742   \tag_mc_end:
1743   \tag_mc_begin_pop:n { }
1744 }
1745 \socket_assign_plug:nm { tagsupport/postnotes/multisep/begin } { default }
1746 \socket_assign_plug:nm { tagsupport/postnotes/multisep/end } { default }
\printpostnotes
1747 \socket_new_plug:nmn { tagsupport/postnotes/printlist/begin } { default }
1748 { \tag_tool:n { para/tagging=false } }
1749 \socket_new_plug:nmn { tagsupport/postnotes/printlist/end } { default }
1750 { }
1751 \socket_assign_plug:nm { tagsupport/postnotes/printlist/begin } { default }
1752 \socket_assign_plug:nm { tagsupport/postnotes/printlist/end } { default }
1753 \socket_new_plug:nmn { tagsupport/postnotes/printnote/begin } { default }
1754 {
1755   \bool_if:NF \l__postnotes_print_as_list_bool
1756   {
1757     \tag_struct_begin:n
1758     {
1759       tag = endnote ,
1760       attribute-class = EndnoteType ,
1761       label = { postnote. \l__postnotes_print_note_id_tl } ,
CHECK Should we really add a back reference here? I couldn't find any hint about this
in the standards, but latex-lab-footnotes does it. No harm, I guess.
1762       ref = { postnotemark. \l__postnotes_print_note_id_tl } ,
1763     }
1764     \__postnotes_tagsup_store_sctructnum:nN
1765     { postnote } \l__postnotes_print_note_id_tl
1766   }
1767 }
1768 \socket_new_plug:nmn { tagsupport/postnotes/printnote/end } { default }
1769 {
1770   \bool_if:NF \l__postnotes_print_as_list_bool
1771   { \tag_struct_end: } % endnote
1772 }
1773 \socket_assign_plug:nm { tagsupport/postnotes/printnote/begin } { default }
1774 \socket_assign_plug:nm { tagsupport/postnotes/printnote/end } { default }
1775 \socket_new_plug:nmn { tagsupport/postnotes/printmark/begin } { default }
1776 {

```

```

1777     \bool_if:NF \l__postnotes_print_as_list_bool
1778     {
1779         \tag_struct_begin:n { tag=endnotelabel }
1780         \tag_mc_begin:n { tag=Lbl }
1781     }
1782 }
1783 \socket_new_plug:nnn { tagsupport/postnotes/printmark/end } { default }
1784 {
1785     \bool_if:NF \l__postnotes_print_as_list_bool
1786     {
1787         \tag_mc_end:
1788         \tag_struct_end: % endnotelabel
1789     }
1790 }
1791 \socket_assign_plug:nn { tagsupport/postnotes/printmark/begin } { default }
1792 \socket_assign_plug:nn { tagsupport/postnotes/printmark/end } { default }
1793 \socket_new_plug:nnn { tagsupport/postnotes/printtext/begin } { default }
1794 {
1795     \bool_if:NTF \l__postnotes_print_as_list_bool
1796     {
1797         \tag_struct_begin:n
1798         {
1799             tag = endnote ,
1800             attribute-class = EndnoteType ,
1801             label = { postnote. \l_postnotes_print_note_id_tl } ,
1802             ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1803         }
1804         \__postnotes_tagsup_store_sstructnum:nN
1805         { postnote } \l_postnotes_print_note_id_tl
1806         \tag_struct_begin:n { tag=text-unit }
1807         \tag_struct_begin:n { tag=text }
1808         \tag_tool:n { para/tagging=true }
1809         \tag_mc_begin:n { }
1810     }
1811     {
1812         \tag_struct_begin:n { tag=text-unit }
1813         \tag_struct_begin:n { tag=text }
1814         \tag_tool:n { para/tagging=true }
1815         \tag_mc_begin:n { }
1816     }
1817 }
1818 \socket_new_plug:nnn { tagsupport/postnotes/printtext/end } { default }
1819 {
1820     \bool_if:NTF \l__postnotes_print_as_list_bool
1821     {
1822         \tag_mc_end:
1823         \tag_tool:n { para/tagging=false }
1824         \tag_struct_end: % text
1825         \tag_struct_end: % text-unit
1826         \tag_struct_end: % endnote
1827     }
1828     {
1829         \tag_mc_end:

```

```

1830         \tag_tool:n { para/tagging=false }
1831         \tag_struct_end: % text
1832         \tag_struct_end: % text-unit
1833     }
1834 }
1835 \socket_assign_plug:mn { tagsupport/postnotes/printtext/begin } { default }
1836 \socket_assign_plug:mn { tagsupport/postnotes/printtext/end } { default }

```

Provide xtemplate based redefinitions of `postnoteslist` and `postnoteslisthang`. This is needed because, as far as I can tell, it is the only way to set `tag` and `attribute-class` for the list struct without tampering with `latex-lab-testphase-block`'s internals.

```

1837 \IfInstanceExistsT { blockenv } { list }
1838 {
1839     \DeclareInstance { blockenv } { postnoteslist } { display }
1840     {
1841         env-name      = postnoteslist ,
1842         tag-name      = L ,
1843         tag-class     = EndnotesList ,
1844         tagging-recipe = list ,
1845         inner-level-counter = ,
1846         level-increase = true ,
1847         setup-code    = ,
1848         block-instance = list ,
1849         inner-instance = postnoteslist ,
1850     }
1851     \DeclareInstanceCopy { blockenv }
1852     { postnoteslisthang } { postnoteslist }
1853     \EditInstance { blockenv } { postnoteslisthang }
1854     { env-name = postnoteslisthang }
1855     \DeclareInstance { list } { postnoteslist } { std }
1856     { item-instance = postnoteslist }
1857     \DeclareInstance { item } { postnoteslist } { std }
1858     {
1859         label-format = { \hspace { \labelsep } \normalfont ~ #1 } ,
1860         label-align = left ,
1861     }
1862     \RenewDocumentEnvironment { postnoteslist } { }
1863     {
1864         \UseInstance { blockenv } { postnoteslist }
1865         {
1866             leftmargin      = Opt ,
1867             label-width     = Opt ,
1868             item-indent     = .5\parindent ,
1869             rightmargin     = Opt ,
1870             parindent       = \parindent ,
1871             par-skip        = \parskip ,
1872             item-skip       = Opt ,
1873             beginsep        = .5\topsep ,
1874             begin-par-skip  = .5\partopsep ,
1875         }
1876     }
1877     { \endblockenv }
1878     \RenewDocumentEnvironment { postnoteslisthang } { }
1879     {

```



```

1880     \UseInstance { blockenv } { postnoteslisthang }
1881     {
1882         leftmargin      = 1em ,
1883         label-width     = -\leftmargin ,
1884         item-indent     = -2\leftmargin ,
1885         rightmargin    = 0pt ,
1886         parindent      = \parindent ,
1887         par-skip       = \parskip ,
1888         item-skip      = 0pt ,
1889         beginsep       = .5\topsep ,
1890         begin-par-skip = .5\partopsep ,
1891     }
1892 }
1893 { \endblockenv }
1894 }

```

Setup for \label and \zlabel inside the note.

```

1895 \bool_new:N \l__postnotes_inside_note_bool
1896 \AddToHookWithArguments { label } [ postnotes/tagsup ]
1897 {
1898     \bool_if:NT \l__postnotes_inside_note_bool
1899     {
1900         \property_record:nn { postnote@label@innote. #1 }
1901         { postnotes/tagsup@noteid }
1902     }
1903 }
1904 \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1905 { \bool_set_true:N \l__postnotes_inside_note_bool }
1906 \property_new:nnnn { postnotes/tagsup@noteid } { now } { 0 }
1907 { \l__postnotes_print_note_id_tl }
1908 \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
1909 {
1910     \zref@newprop { postnotes@tagsup@noteid } [ 0 ]
1911     { \l__postnotes_print_note_id_tl }
1912     \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1913     { \zref@localaddprop { main } { postnotes@tagsup@noteid } }
1914 }

```

Setup for label and zlabel options.

```

1915 \AddToHook { postnotes/note/store } [ postnotes/tagsup ]
1916 {
1917     \str_if_empty:NF \l__postnotes_note_label_str
1918     {
1919         \prop_gput:cnV
1920         { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1921         { label } \l__postnotes_note_label_str
1922     }
1923 }
1924 \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
1925 {
1926     \AddToHook { postnotes/note/store } [ postnotes/tagsup ]
1927     {
1928         \str_if_empty:NF \l__postnotes_note_zlabel_str
1929         {
1930             \prop_gput:cnV

```

```

1931         { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1932         { zlabel } \l__postnotes_note_zlabel_str
1933     }
1934 }
1935 }
1936 \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1937 {
1938     \_postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1939     { label } \l__postnotes_restore_tmp_tl
1940     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1941     {
1942         \exp_args:Ne \property_record:nn
1943         { postnote@label@option. \l__postnotes_restore_tmp_tl }
1944         { postnotes/tagsup@noteid }
1945     }
1946     \_postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1947     { zlabel } \l__postnotes_restore_tmp_tl
1948     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1949     {
1950         \exp_args:Ne \property_record:nn
1951         { postnote@zlabel@option. \l__postnotes_restore_tmp_tl }
1952         { postnotes/tagsup@noteid }
1953     }
1954 }

```

CHECK latex-lab-footnotes creates the footnote structure element (FENote tag) and adds to it a /Ref entry pointing to the structures of *all* marks related to the note, and that includes \footrefs. I don't see anything stating something of the sort in the standards, the backref of the original mark is already a stretch. I also fail to see why this is needed, and how it could be used. But... I trust Ulrike knows better than me.

_postnotes_tagsup_store_crossref:nn

```

\_postnotes_tagsup_store_sstructnum:nN {<ref type>} {<ID number of note>}
\_postnotes_tagsup_store_crossref:nN {<ref type>} {<ID number of note>}
<ref type> is either "postnote" or "postnotemark".

```

```

1955 \prop_new:N \g__postnotes_tagsup_structnums_prop
1956 \cs_new_protected:Npn \_postnotes_tagsup_store_sstructnum:nN #1#2
1957 {
1958     \prop_gput:Nee \g__postnotes_tagsup_structnums_prop
1959     { #1 . #2 } { \tag_get:n { struct_num } }
1960 }
1961 \prop_new:N \g__postnotes_tagsup_crossrefs_prop
1962 \cs_new_protected:Npn \_postnotes_tagsup_store_crossref:nN #1#2
1963 {
1964     \prop_gput:Nee \g__postnotes_tagsup_crossrefs_prop
1965     { \tag_get:n { struct_num } } { #1 . #2 }
1966 }

```

(End of definition for _postnotes_tagsup_store_sstructnum:nN _postnotes_tagsup_store_crossref:nN.)

_postnotes_tagsup_gput_ref:nn

```

\_postnotes_tagsup_gput_ref:nn {<structnum referring from>}
{<structnum being referenced to>}

```

See _fnote_gput_ref:nn.

```

1967 \cs_new_protected:Npn \_postnotes_tagsup_gput_ref:nn #1#2

```

```

1968     {
1969         \tag_if_active:T
1970         { \tag_struct_gput:ene {#1} {ref} { \tag_struct_object_ref:e {#2} } }
1971     }

```

(End of definition for `__postnotes_tagsup_gput_ref:nn`.)

The actual inclusion of the reference has to be done at the end, since the `ref` option called by `\tag_struct_begin:n` does not check if the variable to store the refs already exists, resulting in a clash if we add it immediately and a `\posnoteref` is made before `\printpostnotes`, and also so that the “main” reference always comes first at the list.

```

1972     \AddToHook { tagpdf/finish/before } [ postnotes/tagsup ]
1973     {
1974         \prop_map_inline:Nn \g__postnotes_tagsup_crossrefs_prop
1975         {
1976             \__postnotes_tagsup_gput_ref:nn
1977             { \prop_item:Nn \g__postnotes_tagsup_structnums_prop {#2} }
1978             {#1}
1979         }
1980     }

```

`\postnoteref`

```

1981     \socket_new_plug:nnn { tagssupport/postnotes/postnoteref/begin } { default }
1982     {
1983         \tag_mc_end_push:
1984         \property_if_recorded:eeTF
1985         { postnote@label@innote. \l__postnotes_note_ref_label_tl }
1986         { postnotes/tagsup@noteid }
1987         {

```

Label coming from a `\label` inside the note.

```

1988         \tl_set:Ne \l__postnotes_tmpa_tl
1989         {
1990             \property_ref:ee
1991             { postnote@label@innote. \l__postnotes_note_ref_label_tl }
1992             { postnotes/tagsup@noteid }
1993         }
1994         \tag_struct_begin:n
1995         {
1996             tag = endnotemark ,
1997             ref = { postnote. \l__postnotes_tmpa_tl } ,
1998         }
1999         \__postnotes_tagsup_store_crossref:nN
2000         { postnote } \l__postnotes_tmpa_tl
2001     }
2002     {
2003         \property_if_recorded:eeTF
2004         { postnote@label@option. \l__postnotes_note_ref_label_tl }
2005         { postnotes/tagsup@noteid }
2006         {

```

Label coming from a label option.

```

2007         \tl_set:Ne \l__postnotes_tmpa_tl
2008         {
2009             \property_ref:ee

```

```

2010         { postnote@label@option. \l__postnotes_note_ref_label_tl }
2011         { postnotes/tag-sup@noteid }
2012     }
2013     \tag_struct_begin:n
2014     {
2015         tag = endnotemark ,
2016         ref = { postnotemark. \l__postnotes_tmpa_tl } ,
2017     }
2018     \__postnotes_tag-sup_store_crossref:nN
2019     { postnotemark } \l__postnotes_tmpa_tl
2020 }
2021 { \tag_struct_begin:n { tag = endnotemark } }
2022 }
2023 \tag_mc_begin:n { }
2024 }
2025 \socket_new_plug:nnn { tag-support/postnotes/postnoteref/end } { default }
2026 {
2027     \tag_mc_end:
2028     \tag_struct_end: % endnotemark
2029     \tag_mc_begin_pop:n { }
2030 }
2031 \socket_assign_plug:nn { tag-support/postnotes/postnoteref/begin } { default }
2032 \socket_assign_plug:nn { tag-support/postnotes/postnoteref/end } { default }

```

\postnotezref

```

2033 \AddToHook { package/zref-user/after } [ postnotes/tag-sup ]
2034 {
2035     \socket_new_plug:nnn { tag-support/postnotes/postnotezref/begin } { default }
2036     {
2037         \tag_mc_end_push:
2038         \zref@ifrefcontainsprop { \l__postnotes_note_zref_zlabel_tl }
2039         { postnotes@tag-sup@noteid }
2040         {

```

Label coming from a \zlabel inside the note.

```

2041         \tl_set:Ne \l__postnotes_tmpa_tl
2042         {
2043             \zref@extract { \l__postnotes_note_zref_zlabel_tl }
2044             { postnotes@tag-sup@noteid }
2045         }
2046         \tag_struct_begin:n
2047         {
2048             tag = endnotemark ,
2049             ref = { postnote. \l__postnotes_tmpa_tl } ,
2050         }
2051         \__postnotes_tag-sup_store_crossref:nN
2052         { postnote } \l__postnotes_tmpa_tl
2053     }
2054     {
2055         \property_if_recorded:eeTF
2056         { postnote@zlabel@option. \l__postnotes_note_zref_zlabel_tl }
2057         { postnotes/tag-sup@noteid }
2058         {

```

Label coming from a zlabel option.

```

2059         \tl_set:Nc \l__postnotes_tmpa_tl
2060         {
2061             \property_ref:ee
2062             {
2063                 postnote@zlabel@option.
2064                 \l__postnotes_note_zref_zlabel_tl
2065             }
2066             { postnotes/tag-sup@noteid }
2067         }
2068     \tag_struct_begin:n
2069     {
2070         tag = endnotemark ,
2071         ref = { postnotemark. \l__postnotes_tmpa_tl } ,
2072     }
2073     \__postnotes_tag-sup_store_crossref:nN
2074     { postnotemark } \l__postnotes_tmpa_tl
2075 }
2076 { \tag_struct_begin:n { tag = endnotemark } }
2077 }
2078 \tag_mc_begin:n { }
2079 }
2080 \socket_new_plug:nnn { tag-support/postnotes/postnotezref/end } { default }
2081 {
2082     \tag_mc_end:
2083     \tag_struct_end: % endnotemark
2084     \tag_mc_begin_pop:n { }
2085 }
2086 \socket_assign_plug:nn { tag-support/postnotes/postnotezref/begin } { default }
2087 \socket_assign_plug:nn { tag-support/postnotes/postnotezref/end } { default }
2088 }
2089 }

```

10 Languages

`\pntitle` Set of language specific user variables. They are used in the default value of the `heading` option and in `\pnheaderdefault` which, ultimately, is also used in the same place.

```

\pnhdnotes
\pnhdtopage
\pnhdtopages
2090 \tl_new:N \pntitle
2091 \tl_new:N \pnhdnotes
2092 \tl_new:N \pnhdtopage
2093 \tl_new:N \pnhdtopages
2094 \tl_set:Nn \pntitle { Notes }
2095 \tl_set:Nn \pnhdnotes { Notes }
2096 \tl_set:Nn \pnhdtopage { to~page }
2097 \tl_set:Nn \pnhdtopages { to~pages }

```

(End of definition for `\pntitle` and others.)

`__postnotes_define_language:mn` Defines language specific values for `<postnote language>` by storing a set of assignments for the language specific variables in `<setup>`. `<postnote language>` is an internal name, typically the “main” name of the language, based on which we can set specific `babel` or `polyglossia` languages or variants.

```

\__postnotes_define_language:mn {<postnote language>} {<setup>}

```

```

2098 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
2099 {
2100   \tl_new:c { g__postnotes_language_ #1 _tl }
2101   \tl_gset:cn { g__postnotes_language_ #1 _tl } {#2}
2102 }

```

(End of definition for `__postnotes_define_language:nn`.)

For `babel` we use the new hook system, it's clean, and avoids the `\addto` pitfalls. The appropriate hook to use is `babel/<language>/beforeextras` so that users can override it with a traditional `\addto\extras<language>`.

Note that, for `babel`, the captions are currently handled in two different ways – the “old way” and the “new way” – and which of them is used depends on the language. Most still use the “old way”, but the problem is that it is not universal. And the “new way” uses a different naming scheme – `\<language><caption>`, which is meant to be set with `\setlocalecaption`, and not suitable for our needs. The `\extras<language>` macros are meant for “arbitrary” code to be run when the language is selected, which is what we want. The captions used to work in the same way, but no longer for languages which use the “new way”.

Note also that there seems to exist some qualms about `babel`'s `\addto`. A number of packages define their own versions of it. Do so at least `varioref` (probably the original), `backref`, and `cleveref`. The latter comments that `\addto` is “flawed”. `babel` itself comments the definition recognizing that there is an “inconsistency”: depending on the case, the operation will be either local or global. This is documented in the manual, which explains this inconsistent behavior is preserved for backward compatibility, and recommends `etoolbox`'s facilities if available. `polyglossia` also recommends `etoolbox`'s `\gappto`. All in all, if there's need to use the traditional way instead of the new hooks, just rely on `expl3` and use `\tl_gput_right:Nn`.

`__postnotes_set_babel_language:nn` Sets `<babel language>` to execute the setup defined by `__postnotes_define_language:nn` for `<postnote language>` at the `babel/<language>/beforeextras` hook.

```

\__postnotes_set_babel_language:nn {<babel language>} {<postnote language>}
2103 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
2104 {
2105   \ActivateGenericHook { babel/#1/beforeextras }
2106   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
2107   { g__postnotes_language_ #2 _tl }
2108 }

```

(End of definition for `__postnotes_set_babel_language:nn`.)

`polyglossia` uses a similar set of macros for setting up languages as `babel` does. However, the `\blockextras@<language>` macros are unfortunately internal (despite what the manual says, that's what the code does), thus requiring `\makeatletter/\makeatother` for user configuration, which would be an inconvenience. On the other hand, `polyglossia`'s `\captions<language>` works as in `babel`'s “old way”, meaning it is just a “hook” to which we can append some code. So we use `\captions<language>` for `polyglossia`. Things may complicate here if there's need to set up different values for different language variants, since the hooks available are all necessarily internal, but I doubt we'll ever need variants for these simple strings.

`__postnotes_set_polyglossia_language:nn` Sets *(polyglossia language)* to execute the setup defined by `__postnotes_define_language:nn` for *(postnote language)* at the polyglossia `\captions<language>` hook.

```

    \__postnotes_set_polyglossia_language:nn {(polyglossia language)}
      {(postnote language)}

2109 \cs_new_protected:Npn \__postnotes_set_polyglossia_language:nn #1#2
2110 {
2111   \AddToHook { package/polyglossia/after }
2112   {
2113     \exp_args:Nnv \csgappto { captions #1 }
2114     { g__postnotes_language_ #2 _tl }
2115   }
2116 }
```

(End of definition for __postnotes_set_polyglossia_language:nn.)

English

```

2117 \__postnotes_define_language:nn { english }
2118 {
2119   \tl_set:Nn \pntitle      { Notes }
2120   \tl_set:Nn \pnhdnotes   { Notes }
2121   \tl_set:Nn \pnhdtopage  { to-page }
2122   \tl_set:Nn \pnhdtopages { to-pages }
2123 }
2124 \__postnotes_set_babel_language:nn { english } { english }
2125 \__postnotes_set_babel_language:nn { british } { english }
2126 \__postnotes_set_babel_language:nn { american } { english }
2127 \__postnotes_set_babel_language:nn { canadian } { english }
2128 \__postnotes_set_babel_language:nn { australian } { english }
2129 \__postnotes_set_babel_language:nn { newzealand } { english }
2130 \__postnotes_set_babel_language:nn { UKenglish } { english }
2131 \__postnotes_set_babel_language:nn { USenglish } { english }
2132 \__postnotes_set_polyglossia_language:nn { english } { english }
```

Portuguese

```

2133 \__postnotes_define_language:nn { portuguese }
2134 {
2135   \tl_set:Nn \pntitle      { Notas }
2136   \tl_set:Nn \pnhdnotes   { Notas }
2137   \tl_set:Nn \pnhdtopage  { da-página }
2138   \tl_set:Nn \pnhdtopages { das-páginas }
2139 }
2140 \__postnotes_set_babel_language:nn { portuguese } { portuguese }
2141 \__postnotes_set_babel_language:nn { brazilian } { portuguese }
2142 \__postnotes_set_babel_language:nn { portuges } { portuguese }
2143 \__postnotes_set_babel_language:nn { brazil } { portuguese }
2144 \__postnotes_set_polyglossia_language:nn { portuguese } { portuguese }
```

French

French localization validated by ‘Pika78’ at issue [#1](#).

babel-french also has .ldfs for `francais`, `frenchb`, and `canadien`, but they are deprecated as options and, if used, they fall back to either `french` or `acadian`.

```

2145 \_postnotes_define_language:nn { french }
2146 {
2147   \tl_set:Nn \pntitle      { Notes }
2148   \tl_set:Nn \pnhdnotes   { Notes }
2149   \tl_set:Nn \pnhdtopage  { de-la~page }
2150   \tl_set:Nn \pnhdtopages { des~pages }
2151 }
2152 \_postnotes_set_babel_language:nn { french } { french }
2153 \_postnotes_set_babel_language:nn { acadian } { french }
2154 \_postnotes_set_polyglossia_language:nn { french } { french }

```

German

German localization provided by Herbert Voß at issue [#2](#).

babel-german also has .ldfs for `germanb` and `ngermanb`, but they are deprecated as options and, if used, they fall back respectively to `german` and `ngerman`.

```

2155 \_postnotes_define_language:nn { german }
2156 {
2157   \tl_set:Nn \pntitle      { Endnoten }
2158   \tl_set:Nn \pnhdnotes   { Endnoten }
2159   \tl_set:Nn \pnhdtopage  { zu~Seite }
2160   \tl_set:Nn \pnhdtopages { zu~Seiten }
2161 }
2162 \_postnotes_set_babel_language:nn { german }      { german }
2163 \_postnotes_set_babel_language:nn { ngerman }    { german }
2164 \_postnotes_set_babel_language:nn { austrian }   { german }
2165 \_postnotes_set_babel_language:nn { naustrian }  { german }
2166 \_postnotes_set_babel_language:nn { swissgerman } { german }
2167 \_postnotes_set_babel_language:nn { nswissgerman } { german }
2168 \_postnotes_set_polyglossia_language:nn { german } { german }
2169 </package>

```

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		B	
	1926, 1936, 1972, 2033, 2106, 2111		
<code>\ActivateGenericHook</code>	2105	<code>\AddToHookNext</code>	977
<code>\addto</code>	62	<code>\AddToHookWithArguments</code>	1896
<code>\addtocounter</code>	48		
<code>\AddToHook</code> 112, 133, 354, 454, 1160, 1375, 1400, 1402, 1404, 1406, 1413, 1415, 1427, 1432, 1488, 1492, 1499, 1522, 1530, 1560, 1562, 1569, 1572, 1576, 1583, 1595, 1597, 1606, 1609, 1614, 1624, 1627, 1632, 1643, 1645, 1652, 1656, 1904, 1908, 1912, 1915, 1924,		<code>\begin</code>	903
		<code>\BlockquoteDisable</code>	1612
		bool commands:	
		<code>\bool_do_until:nn</code>	1011
		<code>\bool_gset_false:N</code>	107, 978
		<code>\bool_gset_true:N</code>	83, 832

<code>\bool_if:N</code>	35, 359, 396, 405, 456, 508, 516, 552, 564, 614, 637, 682, 763, 835, 902, 940, 963, 1005, 1092, 1129, 1163, 1380, 1616, 1658, 1755, 1770, 1777, 1785, 1795, 1820, 1898	
<code>\bool_if_exist:N</code>	1654	
<code>\bool_lazy_all:nTF</code>	605, 1541	
<code>\bool_lazy_and:nnTF</code>	96, 537, 734, 808, 988, 1071, 1108, 1140, 1634, 1685	
<code>\bool_lazy_any:nTF</code>	86	
<code>\bool_lazy_or:nnTF</code>	533, 1079	
<code>\bool_lazy_or_p:nn</code>	1115	
<code>\bool_new:N</code>	82, 219, 327, 328, 329, 383, 422, 429, 430, 440, 447, 568, 571, 595, 596, 597, 769, 1377, 1559, 1611, 1626, 1895	
<code>\bool_set_false:N</code>	226, 336, 345, 346, 361, 601, 602, 603	
<code>\bool_set_true:N</code>	231, 335, 340, 341, 579, 1561, 1601, 1602, 1603, 1613, 1618, 1619, 1620, 1629, 1638, 1639, 1640, 1647, 1660, 1661, 1662, 1905	
<code>\bool_to_str:N</code>	51	
<code>\bool_until_do:nn</code>	840	
box commands:		
<code>\box_new:N</code>	20	
<code>\box_use:N</code>	315	
<code>\box_wd:N</code>	314	
C		
<code>\caption</code>	15–17, 42	
<code>\chapter</code>	200	
<code>\citereset</code>	44	
clist commands:		
<code>\clist_map_inline:Nn</code>	1473, 1476, 1479, 1482	
<code>\counterwithin</code>	17	
cs commands:		
<code>\cs_generate_variant:Nn</code>	23, 68, 164, 167, 170, 173, 176, 183, 190, 284, 693, 998, 1374	
<code>\cs_if_eq_p:NN</code>	1636	
<code>\cs_if_exist:N</code>	39, 59, 179, 186, 196, 1401	
<code>\cs_if_exist_p:N</code>	540, 1080, 1116, 1117, 1686	
<code>\cs_new:Npn</code>	21, 74, 184, 272	
<code>\cs_new_protected:Npe</code>	105	
<code>\cs_new_protected:Npn</code>	25, 55, 69, 76, 84, 94, 154, 162, 165, 168, 171, 174, 177, 198, 205, 394, 403, 467, 469, 504, 641, 677, 695, 713, 728, 754, 803, 986, 1000, 1069, 1106, 1127, 1210, 1321, 1378, 1383, 1447, 1465, 1535, 1956, 1962, 1967, 2098, 2103, 2109	
<code>\cs_set:Npe</code>	530, 924	
<code>\cs_set:Npn</code>	529, 921, 1408, 1409	
<code>\cs_set_eq:NN</code>	245, 262, 837, 1485, 1512, 1513, 1630, 1648	
<code>\csgappto</code>	2113	
<code>\csundef</code>	1474, 1477, 1480, 1483	
D		
<code>\DeclareInstance</code>	1839, 1855, 1857	
<code>\DeclareInstanceCopy</code>	1851	
<code>\def</code>	3	
dim commands:		
<code>\dim_compare:nNnTF</code>	407	
<code>\DocumentMetadata</code>	50	
E		
<code>\EditInstance</code>	1853	
<code>\end</code>	965	
<code>\endblockenv</code>	1877, 1893	
<code>\endlist</code>	254, 271	
<code>\endnotemark</code>	16	
<code>\endnotetext</code>	16	
<code>\endrefcontext</code>	44	
<code>\enoteheading</code>	27	
exp commands:		
<code>\exp_args:Ne</code>	665, 1241, 1942, 1950	
<code>\exp_args:NNe</code>	518	
<code>\exp_args:NNNe</code>	622	
<code>\exp_args:NNNo</code>	654	
<code>\exp_args:NNo</code>	654	
<code>\exp_args:Nnv</code>	2106, 2113	
<code>\exp_args:NV</code>	647, 650, 717, 722, 903, 965	
<code>\exp_not:N</code>	107, 108, 109, 110	
<code>\exp_not:n</code>	187	
F		
<code>\fmtversion</code>	5	
fnote internal commands:		
<code>_fnote_gput_ref:nn</code>	58	
<code>\footnote</code>	11, 49	
<code>\footnotemark</code>	11, 15–17	
<code>\footnotesize</code>	305	
<code>\footnotetext</code>	15, 16, 1409	
<code>\footref</code>	58	
fp commands:		
<code>\fp_compare:nNnTF</code>	1148	
<code>\fp_new:N</code>	569	
<code>\fp_set:Nn</code>	578	
<code>\fp_use:N</code>	36	
G		
<code>\gappto</code>	62	

group commands:	
\group_begin:	506, 620, 629, 730, 756, 805, 863, 904, 908, 1002, 1075, 1112, 1131, 1162, 1212, 1323, 1511, 1537
\group_end:	565, 623, 635, 745, 766, 882, 959, 970, 982, 1067, 1101, 1122, 1157, 1177, 1319, 1372, 1516, 1556
	H
\hbox	279
hbox commands:	
\hbox_set:Nn	312
\hspace	273, 1859
\hyperlink	1548
\hyperref	738
	I
\IfFormatAtLeastTF	5, 6
\IfInstanceExistsT	1837
\IfPackageAtLeastTF	1574
\IfPackageLoadedTF	356
\iftoggle	1423
int commands:	
\int_compare:nNnTF	1022, 1052
\int_eval:n	1381
\int_gadd:Nn	100, 470
\int_gincr:N	511, 757, 758, 806
\int_gset:Nn	468
\int_incr:N	621
\int_new:N	80, 494, 498, 753, 786
\int_set:Nn	515, 520, 525, 922, 1436, 1439, 1507, 1587, 1590
\int_use:N	32, 37, 49, 102, 169, 172, 411, 496, 549, 701, 822, 1008, 1386, 1390, 1418, 1420, 1467, 1468, 1495, 1579, 1581
iow commands:	
\iow_char:N	985, 1181
\iow_now:Nn	116, 121, 126, 137, 142, 147
\iow_shipout_e:Nn	6, 158, 476, 486
\item	29, 51, 942
\itemindent	244, 261
\itemsep	249, 266
	K
\kern	398, 399
keys commands:	
\keys_define:nn	191, 212, 220, 274, 288, 297, 330, 363, 385, 423, 431, 441, 448, 458, 572, 770, 1524
\keys_set:nn	492, 507, 761
	L
\label	17, 52, 57, 59, 717, 1512
\labelsep	273, 1859
\labelwidth	243, 260
\lastkern	13, 408
\leftmargin	242, 259, 260, 261, 1883, 1884
\leftskip	307
legacy commands:	
\legacy_if:nTF	114, 135, 156, 474, 484, 1599, 1607
\list	240, 257
\listparindent	247, 264
\lTblrMeasuringBool	50, 1654, 1658
	M
\makeatletter	62
\makeatother	62
\makelabel	245, 262
\MakeLinkTarget	2, 547, 929
\mathchoice	15, 48
\mbox	15
\mkbibendnote	43
mode commands:	
\mode_if_horizontal:TF	698, 709
\mode_leave_vertical:	697, 947, 950
msg commands:	
\msg_line_context:	378, 985, 1105, 1126, 1182
\msg_new:nnn	377, 379, 984, 1104, 1125, 1179
\msg_warning:nn	360, 824, 1121
\msg_warning:nnn	367, 372, 462, 1094, 1174
\multfootsep	12
\multiplefootnotemarker	11, 12
	N
\NeedsTeXFormat	4
\newcounter	493, 800, 801
\NewDocumentCommand	471, 481, 491, 501, 725, 747, 778, 1393, 1532
\NewDocumentEnvironment	238, 255
\NewHook	3, 24, 503, 598, 798, 799
\newlabel	4
\NewTblrEnviron	50
\nobreak	703
\noindent	322
\normalfont	273, 279, 313, 323, 1859
	P
\PackageError	9
\par	30, 227, 966, 969
\parindent	244, 247, 264, 308, 1868, 1870, 1886

<code>\parsep</code>	248, 265	<code>__postnotes_biblatex_citereset_-</code>	
<code>\parskip</code>	248, 265, 1871, 1887	<code>local:</code>	1430, 1465, 1465
<code>\partopsep</code>	251, 268, 1874, 1890	<code>__postnotes_biblatex_endrefcontext_-</code>	
<code>\pnaddtocounter</code> aux	14, 467	<code>local:</code>	1429, 1447, 1447
<code>\pnhdchapfirst</code> ...	1184, 1215, 1338, 1339	<code>\l__postnotes_biblatex_orig_-</code>	
<code>\pnhdchaplast</code> ...	1184, 1216, 1343, 1347	<code>refsection_tl</code> .	45, 1490, 1494, 1497
<code>\pnhdnamefirst</code> ...	1184, 1219, 1362, 1363	<code>\g__postnotes_biblatex_prev_-</code>	
<code>\pnhdnamelast</code> ...	1184, 1220, 1367, 1371	<code>refsection_tl</code>	1491, 1496, 1505, 1509
<code>\pnhdnotes</code>	1396,	<code>\l__postnotes_check_dupli_bool</code> ..	
	1397, 2090, 2120, 2136, 2148, 2158	429, 433, 1092
<code>\pnhdpagefirst</code>	1184,	<code>__postnotes_check_duplicates:N</code> .	
	1213, 1326, 1327, 1395, 1396, 1397	33, 829, 1069, 1069
<code>\pnhdpagelast</code>		<code>__postnotes_check_floats:N</code>	
..	1184, 1214, 1331, 1335, 1395, 1397	31, 34, 831, 1106, 1106
<code>\pnhdsectfirst</code> ...	1184, 1217, 1350, 1351	<code>\l__postnotes_check_floats_bool</code> .	
<code>\pnhdsectlast</code> ...	1184, 1218, 1355, 1359	430, 436, 1109
<code>\pnhdtopage</code>		<code>\l__postnotes_clear_queue_seq</code> ...	
..	1396, 2090, 2121, 2137, 2149, 2159	786, 819, 979
<code>\pnhdtopages</code>		<code>\g__postnotes_countersaux_bool</code> ...	
..	1397, 2090, 2122, 2138, 2150, 2160	31, 97, 447, 450,
<code>\pnheaderdefault</code> ..	42, 61, 201, 208, 1393		456, 516, 534, 809, 1005, 1073, 1163
<code>\pnheading</code>	7, 193, 196, 826	<code>\g__postnotes_countersaux_prop</code> ...	
<code>\pnidnextnote</code>	780, 870	81, 101, 518
<code>\pnsetcounteraux</code>	14, 467	<code>\l__postnotes_countersaux_step_-</code>	
<code>\pnthechapter</code>	780, 866	<code>int</code>	494, 515, 525, 549
<code>\pnthechapternextnote</code>	780, 874	<code>\l__postnotes_csquotes_measuring_-</code>	
<code>\pnthepage</code>	780, 910	<code>bool</code>	1611, 1613, 1616
<code>\pnthesection</code>	780, 869	<code>\l__postnotes_curr_text_page_tl</code> .	
<code>\pnthesectionnextnote</code>	780, 877	39, 1192, 1234, 1246, 1247,
<code>\pntitle</code>			1251, 1279, 1282, 1285, 1288, 1299
	200, 207, 2090, 2119, 2135, 2147, 2157	<code>__postnotes_data_name:n</code>	
<code>\posnoteref</code>	59	2, 17, 21, 21, 23, 27, 28, 29,
<code>\postnote</code>	2, 3, 15, 17–19, 23,		31, 33, 41, 44, 46, 48, 50, 52, 57, 58,
	33, 36, 42, 43, 48, 50, 52, 53, 501, 1408		61, 64, 66, 71, 75, 77, 1417, 1419,
<code>\postnotemark</code>	16, 17		1421, 1424, 1578, 1580, 1920, 1931
<code>\postnoteref</code>	17, 23, 48, 52, 59, 725	<code>__postnotes_define_language:nm</code> .	
postnotes commands:		61–
<code>\c__postnotes_multi_notemarker_tl</code>			63, 2098, 2098, 2117, 2133, 2145, 2155
.....	382, 398, 399, 408	<code>__postnotes_extract_pageref:n</code> ..	
<code>\l__postnotes_note_id_tl</code>	17,	7, 177, 184, 190, 1280, 1389
	494, 519, 528, 531, 541, 547, 548,	<code>\g__postnotes_firstrun_bool</code>	
	560, 759, 762, 764, 765, 1417, 1419,	82, 83, 107, 538, 810, 1072, 1110
	1421, 1424, 1578, 1580, 1705, 1706,	<code>__postnotes_get_headers_data:N</code> .	
	1709, 1723, 1724, 1727, 1920, 1931	36, 37, 40, 833, 1210, 1210
<code>\l__postnotes_print_note_id_tl</code> ...		<code>__postnotes_get_label_if_-</code>	
.....	52, 786, 843, 844, 865, 868,	<code>exist:N</code>	21, 543, 626, 641, 641
	879, 911, 913, 916, 919, 930, 932,	<code>__postnotes_get_pageref:Nn</code> ..	7,
	935, 1434, 1437, 1440, 1443, 1501,		177, 177, 183, 910, 1245, 1253, 1290
	1585, 1588, 1761, 1762, 1765, 1801,	<code>\g__postnotes_header_chap_first_-</code>	
	1802, 1805, 1907, 1911, 1938, 1946	<code>prop</code>	1192, 1223, 1281, 1336
postnotes internal commands:		<code>\g__postnotes_header_chap_last_-</code>	
<code>\l__postnotes_backlink_bool</code>		<code>prop</code> ...	1192, 1224, 1268, 1309, 1340
.....	329, 350, 990	<code>\g__postnotes_header_name_first_-</code>	
		<code>prop</code>	1192, 1227, 1287, 1360

\g__postnotes_header_name_last_-
prop ... 1192, 1228, 1274, 1315, 1364
\g__postnotes_header_page_first_-
prop 1192, 1221, 1278, 1324
\g__postnotes_header_page_last_-
prop ... 1192, 1222, 1265, 1306, 1328
\g__postnotes_header_prev_last_-
chap_tl 1192, 1230, 1339, 1344, 1347
\g__postnotes_header_prev_last_-
name_tl 1192, 1232, 1363, 1368, 1371
\g__postnotes_header_prev_last_-
page_tl 1192, 1229, 1327, 1332, 1335
\g__postnotes_header_prev_last_-
sect_tl 1192, 1231, 1351, 1356, 1359
\g__postnotes_header_sect_first_-
prop 1192, 1225, 1284, 1348
\g__postnotes_header_sect_last_-
prop ... 1192, 1226, 1271, 1312, 1352
\g__postnotes_header_vars_next_-
bool 41, 832, 978, 1377, 1380
\l__postnotes_hyperlink_bool 327,
335, 340, 345, 361, 682, 736, 989, 1544
\l__postnotes_hyperref_warn_bool
..... 328, 336, 341, 346, 359
__postnotes_inhibit_note: ... 599
__postnotes_inhibit_note:TF ...
..... 19, 33, 509, 595
\l__postnotes_inhibit_note_bool .
..... 20, 595,
601, 607, 637, 1601, 1618, 1638, 1660
\l__postnotes_inside_note_bool .
..... 1895, 1898, 1905
\g__postnotes_labelseq_seq
..... 31, 79, 92, 1009, 1010,
1014, 1039, 1040, 1043, 1065, 1165
__postnotes_list_makelabel:n ...
..... 245, 262, 272
__postnotes_make_mark:nnn
..... 12, 276, 284,
415, 633, 684, 688, 739, 741, 1550, 1552
__postnotes_make_text_mark:nnn .
..... 280, 992, 996
\l__postnotes_manual_sortnum_-
bool 35, 571, 579
\l__postnotes_mark_tl
..... 21, 30, 512, 523,
530, 532, 567, 574, 612, 628, 724, 1529
\l__postnotes_mark_typeset_tl ...
..... 494,
532, 543, 560, 617, 624, 626, 628, 633
\l__postnotes_maybe_multi_bool .
..... 33, 51, 440, 443, 535, 592
\l__postnotes_multiple_bool
..... 383, 387, 396, 405
__postnotes_multiple_check: ...
..... 13, 403, 702
__postnotes_multiple_prepare: ..
..... 12, 394, 708
\l__postnotes_multisep_tl
..... 381, 390, 415
\l__postnotes_nomark_bool
..... 508, 552, 564, 568, 588, 609
__postnotes_note:nn
..... 18, 22, 23, 502, 503, 504
\g__postnotes_note_id_int
.... 17, 31, 494, 511, 758, 1025, 1055
\l__postnotes_note_label_str ...
..... 570, 590,
643, 666, 672, 715, 717, 718, 1917, 1921
__postnotes_note_ref:nn
..... 23, 726, 727, 728
\l__postnotes_note_ref_label_tl .
.... 727, 731, 1985, 1991, 2004, 2010
\l__postnotes_note_set_labels_tl
..... 494, 545, 555, 561
\l__postnotes_note_zlabel_str ...
..... 46, 645, 647, 651,
657, 721, 722, 1521, 1526, 1928, 1932
__postnotes_note_zref:nn
..... 47, 1533, 1534, 1535
\l__postnotes_note_zref_zlabel_-
tl 1534, 1538, 2038, 2043, 2056, 2064
\l__postnotes_post_printnote_tl .
..... 227, 287, 294, 958
\l__postnotes_post_textmark_tl ..
..... 286, 292, 937
\g__postnotes_postnote_counteraux_-
int 80, 100, 102, 468, 470
\l__postnotes_pre_textmark_tl ...
..... 285, 290, 933
\l__postnotes_prev_mark_chap_tl .
.. 1192, 1236, 1256, 1270, 1293, 1311
\l__postnotes_prev_mark_name_tl .
.. 1192, 1238, 1260, 1276, 1297, 1317
\l__postnotes_prev_mark_page_tl .
.. 1192, 1235, 1254, 1267, 1291, 1308
\l__postnotes_prev_mark_sect_tl .
.. 1192, 1237, 1258, 1273, 1295, 1314
\l__postnotes_prev_text_page_tl .
..... 39, 1192, 1233,
1250, 1263, 1266, 1269, 1272, 1275,
1298, 1304, 1307, 1310, 1313, 1316
\l__postnotes_print_as_list_bool
.... 219, 226, 231, 835, 902, 940,
963, 1755, 1770, 1777, 1785, 1795, 1820
\l__postnotes_print_content_tl ..
..... 786, 880, 881, 920, 955

\l__postnotes_print_counter_tl 786, 917, 923
\l__postnotes_print_env_tl 218, 228, 232, 903, 965
\l__postnotes_print_format_tl 211, 214, 906
\g__postnotes_print_labelseq_
queue_seq 813, 999, 1036, 1063, 1120
\l__postnotes_print_mark_tl 786, 914, 925, 936
\l__postnotes_print_note_id_
next_tl 786, 850,
855, 857, 871, 873, 876, 890, 895, 897
__postnotes_print_notes: 25, 26, 30, 35, 36, 779, 803, 803
\l__postnotes_print_plain_mark_
bool 20,
596, 602, 608, 1602, 1619, 1639, 1661
\l__postnotes_print_plain_mark_
stepcounter_bool 20, 50,
597, 603, 614, 1603, 1620, 1640, 1662
\g__postnotes_print_postnotes_
int . 786, 806, 822, 1008, 1386, 1390
\g__postnotes_print_queue_seq 26, 33–35, 37,
786, 812, 816, 819, 823, 829, 830,
831, 833, 840, 842, 848, 854, 888, 894
\l__postnotes_print_type_curr_tl 786, 845, 846, 884, 974
\l__postnotes_print_type_next_tl 786, 851, 858, 860, 891, 898, 960
\l__postnotes_print_type_prev_tl 786, 828, 883, 900, 973
\l__postnotes_print_typeset_
mark_tl 786, 926, 945, 952
__postnotes_prop_gclear:n 4, 69, 76, 980
__postnotes_prop_get:nnN 4, 69, 69, 844,
856, 864, 867, 872, 875, 878, 896,
912, 915, 918, 1134, 1135, 1138,
1139, 1144, 1146, 1255, 1257, 1259,
1292, 1294, 1296, 1434, 1437, 1440,
1443, 1501, 1585, 1588, 1938, 1946
__postnotes_prop_item:nn 4, 69, 74, 1083, 1088,
1095, 1170, 1242, 1283, 1286, 1289
\g__postnotes_queue_seq 17, 494, 527, 759, 817, 820, 1169
\c__postnotes_ref_prefix_tl 4, 78, 110, 179,
180, 186, 187, 541, 1080, 1116, 1117
\l__postnotes_restore_tmp_tl 1399, 1435, 1436, 1438, 1439,
1441, 1442, 1444, 1445, 1502, 1504,
1508, 1510, 1586, 1587, 1589, 1590,
1939, 1940, 1943, 1947, 1948, 1951
\l__postnotes_saved_spacefactor_
multi_tl 384, 410, 417
\l__postnotes_saved_spacefactor_
tl 694, 700, 710
\g__postnotes_sectid_int 49, 753, 757
__postnotes_section:nn 24, 750, 753, 754
\l__postnotes_section_exp_bool 763, 769, 774
\g__postnotes_section_name_tl 47, 760, 768, 772
__postnotes_set_babel_language:nn 62, 2103, 2103, 2124, 2125,
2126, 2127, 2128, 2129, 2130, 2131,
2140, 2141, 2142, 2143, 2152, 2153,
2162, 2163, 2164, 2165, 2166, 2167
__postnotes_set_headers_vars:n 37, 40, 41, 1321, 1321, 1374, 1381, 1387
__postnotes_set_headers_vars_
first: 27, 37, 41, 834, 1375, 1383
__postnotes_set_headers_vars_
next: 37, 41, 1375, 1376, 1378
__postnotes_set_label:nnnn 6, 154, 154, 163, 166, 169, 172, 175
__postnotes_set_mark_page_
label:nn 6, 36, 154, 162, 164, 548
__postnotes_set_polyglossia_
language:nn 63,
2109, 2109, 2132, 2144, 2154, 2168
__postnotes_set_pre_print_
label:n 6, 154, 174, 176, 821
__postnotes_set_print_page_
label:n 6, 27, 36, 154, 171, 173, 1385
__postnotes_set_section_page_
label:n 6, 154, 165, 167, 762
__postnotes_set_text_page_
label:n 6, 36, 154, 168, 170, 931
__postnotes_set_user_labels: 46, 550, 713, 713
\l__postnotes_sort_bool 422, 425, 1129
\l__postnotes_sort_num_fp 36, 569, 578
__postnotes_sort_queue:N 35, 830, 1127, 1127
__postnotes_split_labelseq: 807, 999, 1000
__postnotes_step_counteraux:nnn 4, 78, 94, 109
__postnotes_store:nn . 3, 24, 25, 531
__postnotes_store_labelseq:nn 4, 78, 84, 108

_postnotes_store_section:nn ...	prg commands:
..... 3, 55, 55, 68, 764, 765	\prg_new_protected_conditional:Npnn
\l_postnotes_tabularx_inside_- 599
env_bool ... 1626, 1629, 1635, 1647	\prg_return_false: 639
_postnotes_tabularx_saved_-	\prg_return_true: 638
write:Nn 1630, 1636, 1648	\printpostnotes ... 17, 25, 26, 30-33,
\g_postnotes_tagsup_crossrefs_-	36, 37, 39-41, 44, 51, 52, 54, 59, 778
prop 1961, 1964, 1974	prop commands:
_postnotes_tagsup_gput_ref:nn .	\prop_gclear:N ... 77, 1221, 1222,
..... 58, 1967, 1967, 1976	1223, 1224, 1225, 1226, 1227, 1228
_postnotes_tagsup_store_-	\prop_get:NnNTF .. 71, 1324, 1328,
crossref:nN 58, 1962, 1999, 2018, 2051, 2073	1336, 1340, 1348, 1352, 1360, 1364
_postnotes_tagsup_store_-	\prop_gpop:NnNTF 518
sctructnum:nN 58, 1708, 1726, 1764, 1804, 1956	\prop_gput:Nnn 28, 29,
_postnotes_tagsup_store_-	31, 33, 41, 44, 46, 48, 50, 52, 58, 61,
sctructnum:nN_postnotes_-	64, 66, 101, 1265, 1268, 1271, 1274,
tagsup_store_crossref:nN ... 1955	1278, 1281, 1284, 1287, 1306, 1309,
\g_postnotes_tagsup_structnums_-	1312, 1315, 1417, 1419, 1421, 1424,
prop 1955, 1958, 1977	1578, 1580, 1919, 1930, 1958, 1964
\l_postnotes_tmpa_box 16, 312, 314, 315	\prop_item:Nn 75, 1977
\l_postnotes_tmpa_seq 16, 1003, 1031, 1037, 1038, 1040,	\prop_map_inline:Nn 1974
1058, 1064, 1076, 1085, 1091, 1100,	\prop_new:N 27, 57, 81, 1192, 1193, 1194, 1195,
1113, 1120, 1165, 1169, 1172, 1175	1196, 1197, 1198, 1199, 1955, 1961
\l_postnotes_tmpa_tl 16, 519, 520, 521,	property commands:
1007, 1009, 1010, 1012, 1134, 1136,	\property_if_recorded:nnTF 665, 1984, 2003, 2055
1138, 1141, 1145, 1149, 1325, 1326,	\property_new:nnnn 724, 1906
1329, 1331, 1333, 1337, 1338, 1341,	\property_record:nn 718, 1900, 1942, 1950
1343, 1345, 1349, 1350, 1353, 1355,	\property_ref:nn 671, 1990, 2009, 2061
1357, 1361, 1362, 1365, 1367, 1369,	\providecommand 5, 118, 123, 128, 139, 144, 149
1988, 1997, 2000, 2007, 2016, 2019,	\ProvidesExplPackage 14
2041, 2049, 2052, 2059, 2071, 2074	
\l_postnotes_tmpb_seq 16, 1004, 1027, 1038, 1056, 1065	R
\l_postnotes_tmpb_tl 16, 1012, 1014, 1016, 1023, 1028, 1032,	\ref 2, 739, 741
1135, 1136, 1139, 1142, 1147, 1149	\refsection 45, 1515
_postnotes_typeset_mark:nnN ... 22, 559, 677, 677, 693	\refstepcounter 17
_postnotes_typeset_mark_-	\RenewDocumentEnvironment ... 1862, 1878
wrapper:nnn 22, 632, 677, 679, 695, 732, 1539	\rightmargin 246, 263
_postnotes_typeset_text_-	\rightskip 306
mark:nn 30, 934, 986, 986, 998	S
\l_postnotes_zrefhyperref_bool .	scan commands:
..... 1545, 1559, 1561	\scan_stop: 400, 418, 711
\postnotessection 3, 24, 25, 27, 747	\section 207
\postnotesetup .. 15, 457, 491, 1401, 1607	seq commands:
\postnotetext 16, 17	\seq_clear:N 1003, 1004, 1076
\postnotezref 47, 52, 60, 1532	\seq_concat:NNN 1038
	\seq_count:N 1175
	\seq_gclear:N 820
	\seq_get_left:NN 854, 894
	\seq_gpop_left:NN 842, 1014

<code>\seq_gput_right:Nn</code>	92, 527, 759, 1010	<code>\tag_if_active:TF</code>	1969
<code>\seq_gset_eq:NN</code>	812, 816, 1036, 1040, 1063, 1065, 1100	<code>\tag_if_active_p:</code>	1686, 1687
<code>\seq_gsort:Nn</code>	1132	<code>\tag_mc_begin:n</code>	1710, 1738, 1780, 1809, 1815, 2023, 2078
<code>\seq_if_empty:NTF</code>	823, 848, 888, 1172	<code>\tag_mc_begin_pop:n</code>	1716, 1743, 2029, 2084
<code>\seq_if_empty_p:N</code>	840	<code>\tag_mc_end:</code>	1714, 1742, 1787, 1822, 1829, 2027, 2082
<code>\seq_if_eq:NNTF</code>	35	<code>\tag_mc_end_push:</code>	1701, 1737, 1983, 2037
<code>\seq_if_in:NnTF</code>	1009	<code>\tag_socket_use:n</code>	414, 416, 554, 556, 690, 691, 743, 744, 905, 928, 938, 944, 951, 954, 956, 957, 962, 1554, 1555
<code>\seq_map_inline:Nn</code>	979, 1043, 1077, 1239	<code>\tag_struct_begin:n</code>	59, 1702, 1720, 1757, 1779, 1797, 1806, 1807, 1812, 1813, 1994, 2013, 2021, 2046, 2068, 2076
<code>\seq_new:N</code>	18, 19, 79, 497, 787, 797, 999	<code>\tag_struct_end:</code>	1715, 1730, 1771, 1788, 1824, 1825, 1826, 1831, 1832, 2028, 2083
<code>\seq_put_right:Nn</code>	1027, 1031, 1056, 1058, 1085, 1091	<code>\tag_struct_gput:nnn</code>	1970
<code>\seq_set_eq:NN</code>	819	<code>\tag_struct_object_ref:n</code>	1970
<code>\seq_set_filter:NNn</code>	1113, 1165, 1169	<code>\tag_tool:n</code>	1748, 1808, 1814, 1823, 1830
<code>\setcounter</code>	48, 802	<code>\tagpdfsetup</code>	1689
<code>\setlength</code>	242, 243, 244, 246, 247, 248, 249, 250, 251, 259, 260, 261, 263, 264, 265, 266, 267, 268, 306, 307, 308	<code>T_EX</code> and <code>L^AT_EX_{2ϵ}</code> commands:	
<code>\setlocalecaption</code>	62	<code>\@afterindentfalse</code>	837
skip commands:		<code>\@afterindenttrue</code>	27, 837, 838
<code>\skip_horizontal:n</code>	314	<code>\@auxout</code>	158, 476, 486
<code>\small</code>	215	<code>\@bsphack</code>	473, 483, 508, 749
socket commands:		<code>\@capttype</code>	1401
<code>\socket_assign_plug:nn</code>	630, 631, 1731, 1732, 1733, 1734, 1745, 1746, 1751, 1752, 1773, 1774, 1791, 1792, 1835, 1836, 2031, 2032, 2086, 2087	<code>\@currentcounter</code>	529, 921
<code>\socket_new:nn</code>	1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684	<code>\@currentlabel</code>	530, 924
<code>\socket_new_plug:nnn</code>	1699, 1712, 1718, 1729, 1735, 1740, 1747, 1749, 1753, 1768, 1775, 1783, 1793, 1818, 1981, 2025, 2035, 2080	<code>\@esphack</code>	479, 489, 564, 751
sort commands:		<code>\@footnotemark</code>	22
<code>\sort_return_same:</code>	1151, 1153, 1155	<code>\@ifl@t@r</code>	5
<code>\sort_return_swapped:</code>	1150	<code>\@mainaux</code>	116, 121, 126, 137, 142, 147
<code>\space</code>	11	<code>\@makecaption</code>	42
<code>\spacefactor</code>	411, 417, 701, 710	<code>\@makefnmark</code>	9
<code>\stepcounter</code>	17, 514, 616, 862	<code>\@mkboth</code>	201, 208
str commands:		<code>\@newl@bel</code>	4, 110
<code>\str_case:nnTF</code>	32, 1015, 1045	<code>\@textsuperscript</code>	279, 313
<code>\str_if_empty:NTF</code>	643, 645, 715, 721, 1917, 1928	<code>\blx@bibfiles</code>	45
<code>\str_if_eq:nnTF</code>	1087	<code>\blx@edef@refcontext</code>	1445, 1456
<code>\str_if_eq_p:nn</code>	89, 90, 98, 1082, 1141, 1142, 1166, 1170	<code>\blx@endrefsection</code>	1514
<code>\str_new:N</code>	570, 1521	<code>\blx@err@endnote</code>	1409
		<code>\blx@info</code>	1513
		<code>\blx@lasthash@foot</code>	1472
		<code>\blx@lasthash@text</code>	1471
		<code>\blx@lastkey@foot</code>	1470
		<code>\blx@lastkey@text</code>	1469
		<code>\blx@lastmpfn</code>	1485
		<code>\blx@refcontext@context</code>	1425
		<code>\blx@refcontext@labelalphametemplate</code>	1455, 1462
tag commands:			
<code>\tag_get:n</code>	1959, 1965		

<code>\blx@refcontext@labelprefix</code> ..	1450	tl commands:
<code>\blx@refcontext@labelprefix@real</code>		<code>\c_empty_tl</code>
.....	1451	<code>\tl_clear:N</code> 72, 181, 521, 1233, 1234,
<code>\blx@refcontext@sortingnamekeytemplatenam</code>		1235, 1236, 1237, 1238, 1450, 1451,
.....	1453, 1459	1467, 1468, 1475, 1478, 1481, 1484
<code>\blx@refcontext@sortingtemplatenam</code>		<code>\tl_const:Nn</code>
.....	1452, 1458	78, 382
<code>\blx@refcontext@uniquenametemplatenam</code>		<code>\tl_gclear:N</code>
.....	1454, 1461	760,
<code>\blx@sorting</code>	1452	1213, 1214, 1215, 1216, 1217, 1218,
<code>\blx@theendnote</code>	1408	1219, 1220, 1229, 1230, 1231, 1232
<code>\blx@theendnotetext</code>	1409	<code>\tl_gput_right:Nn</code>
<code>\blx@trackhash@foot</code>	1476, 1478	62
<code>\blx@trackhash@text</code>	1473, 1475	<code>\tl_gset:Nn</code> 1326, 1327, 1331, 1332,
<code>\blx@trackkeys@foot</code>	1482, 1484	1335, 1338, 1339, 1343, 1344, 1347,
<code>\blx@trackkeys@text</code>	1479, 1481	1350, 1351, 1355, 1356, 1359, 1362,
<code>\c@blx@maxsection</code>	1507	1363, 1367, 1368, 1371, 1496, 2101
<code>\c@page</code>	169, 172, 1381	<code>\tl_gset_eq:NN</code>
<code>\c@postnote</code>	20, 32, 37, 520, 621	1509
<code>\c@postnotetext</code>	922	<code>\tl_if_empty:NTF</code>
<code>\c@refsection</code>		512, 612, 1247, 1263, 1304, 1940, 1948
.....	1418, 1436, 1467, 1468, 1495	<code>\tl_if_eq:NNTF</code>
<code>\c@refsegment</code>	1420, 1439	1119, 1136, 1249, 1395, 1503
<code>\c@z@abschap</code>	1579, 1587	<code>\tl_if_eq:NnTF</code>
<code>\c@z@abssec</code>	1581, 1590	846, 860, 900, 960
<code>\FN@mf@check</code>	12	<code>\tl_if_eq:nTF</code>
<code>\FN@mf@prepare</code>	11, 12	224, 1241
<code>\hyper@linkend</code>	686, 994	<code>\tl_if_eq_p:NN</code>
<code>\hyper@linkstart</code>	685, 993	1012
<code>\ifmeasuring@</code>	48	<code>\tl_item:Nn</code>
<code>\p@postnote</code>	530, 925	1016, 1023, 1032
<code>\post@note</code>	4, 5, 78, 119, 140, 159	<code>\tl_item:nn</code> ...
<code>\postnote@addtocounter@aux</code>		1046, 1053, 1059, 1166
.....	5, 14, 129, 150, 467	<code>\tl_new:N</code>
<code>\postnote@setcounter@aux</code>
.....	5, 14, 124, 145, 467	16, 17, 211, 218, 285, 286, 287,
<code>\postnotes@required@kernel</code>	3, 4, 6, 11	381, 384, 495, 499, 500, 567, 694,
<code>\z@</code>	1485	727, 768, 780, 781, 782, 783, 784,
<code>\zref@extract</code>	656, 2043	785, 788, 789, 790, 791, 792, 793,
<code>\zref@extractdefault</code>	1549	794, 795, 796, 1184, 1185, 1186,
<code>\zref@ifrefcontainsprop</code> ...	650, 2038	1187, 1188, 1189, 1190, 1191, 1200,
<code>\zref@ifrefundefined</code>	647	1201, 1202, 1203, 1204, 1205, 1206,
<code>\zref@localaddprop</code>	1531, 1913	1207, 1208, 1209, 1399, 1490, 1491,
<code>\zref@newprop</code>	1529, 1910	1534, 2090, 2091, 2092, 2093, 2100
<code>\text</code>	15–17, 48	<code>\tl_set:Nn</code>
<code>\textnormal</code>	48	180, 227, 228, 232,
<code>\textsuperscript</code>	12	410, 496, 523, 545, 617, 624, 654,
<code>\textup</code>	48	669, 700, 731, 828, 850, 851, 870,
<code>\thechapter</code>	36, 42, 62	883, 890, 891, 926, 973, 1007, 1298,
<code>\theHpostnote</code>	17	1452, 1453, 1454, 1455, 1494, 1538,
<code>\thepage</code>	36, 163, 166	1988, 2007, 2041, 2059, 2094, 2095,
<code>\thepostnote</code>	20, 523, 617, 624	2096, 2097, 2119, 2120, 2121, 2122,
<code>\thesection</code>	36, 45, 65	2135, 2136, 2137, 2138, 2147, 2148,
		2149, 2150, 2157, 2158, 2159, 2160
		<code>\tl_set_eq:NN</code>
		532, 628
		<code>\togglefalse</code>
		1449
		<code>\toggletrue</code>
		1405
		token commands:
		<code>\token_to_str:N</code>
		..
		118, 119, 123, 124, 128, 129, 139,
		140, 144, 145, 149, 150, 159, 477, 487
		<code>\topsep</code>
		250, 267, 1873, 1889
		U
		<code>\undef</code>
		1469, 1470, 1471, 1472

<code>\unkern</code>	412, 413		
use commands:			
<code>\use:N</code>	1442		
<code>\use_none:n</code>	1512, 1513		
<code>\UseHook</code>	53, 526, 604, 827, 909		
<code>\UseInstance</code>	1864, 1880		
		W	
		<code>\write</code>	33, 1630, 1636, 1648
		Z	
		<code>\zcsetup</code>	1564, 1570
		<code>\zlabel</code>	52, 57, 60, 722
		<code>\zref</code>	1550, 1552