

# The `postnotes` package

## Code documentation

`gusbrs`

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

Version v0.4.2 – 2024-11-27

### Contents

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

# 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} {2024-11-27} {0.4.2}
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:cnV { \__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       \protected@edef \l__postnotes_tmpa_tl { \thechapter }  
42       \prop_gput:cnV { \__postnotes_data_name:e {#1} }  
43         { thechapter } \l__postnotes_tmpa_tl  
44     }  
45   \protected@edef \l__postnotes_tmpa_tl { \thesection }  
46   \prop_gput:cnV { \__postnotes_data_name:e {#1} } { thesection }  
47     \l__postnotes_tmpa_tl  
48   \prop_gput:cnV { \__postnotes_data_name:e {#1} } { pnsectname }  
49     \g__postnotes_section_name_tl  
50   \prop_gput:cne { \__postnotes_data_name:e {#1} } { pnsectid }  
51     { \int_use:N \g__postnotes_sectid_int }  
52   \prop_gput:cne { \__postnotes_data_name:e {#1} } { multibool }  
53     { \bool_to_str:N \l__postnotes_maybe_multi_bool }  
54   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}  
55   \tl_clear:N \l__postnotes_tmpa_tl  
56   \UseHook { postnotes/note/store }  
57 }
```

(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 $\rangle$ }
```

```

58 \cs_new_protected:Npn \__postnotes_store_section:nn #1#2
59 {
60   \prop_new:c { \__postnotes_data_name:e {#1} }
61   \prop_gput:cn { \__postnotes_data_name:e {#1} } { type } { section }
62   \cs_if_exist:cT { chapter }
63   {
64     \protected@edef \l__postnotes_tmpa_tl { \thechapter }
65     \prop_gput:cnV { \__postnotes_data_name:e {#1} }
66     { thechapter } \l__postnotes_tmpa_tl
67   }
68   \protected@edef \l__postnotes_tmpa_tl { \thesection }
69   \prop_gput:cnV { \__postnotes_data_name:e {#1} } { thesection }
70   \l__postnotes_tmpa_tl
71   \prop_gput:cn { \__postnotes_data_name:e {#1} } { content } {#2}
72   \tl_clear:N \l__postnotes_tmpa_tl
73 }
74 \cs_generate_variant:Nn \__postnotes_store_section:nn { nV }

```

(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>}

```

```

75 \cs_new_protected:Npn \__postnotes_prop_get:nnN #1#2#3
76 {
77   \prop_get:cnNF { \__postnotes_data_name:e {#1} } {#2} #3
78   { \tl_clear:N #3 }
79 }
80 \cs_new:Npn \__postnotes_prop_item:nn #1#2
81 { \prop_item:cn { \__postnotes_data_name:e {#1} } {#2} }
82 \cs_new_protected:Npn \__postnotes_prop_gclear:n #1
83 { \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` `\post@note` is the `\newlabel` equivalent for postnotes. Based on the kernel's `\@newl@bel` so that we get L<sup>A</sup>T<sub>E</sub>X 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>}

```

```

84 \tl_const:Nn \c__postnotes_ref_prefix_tl { postnote@r }
85 \seq_new:N \g__postnotes_labelseq_seq
86 \int_new:N \g__postnotes_postnote_countersaux_int
87 \prop_new:N \g__postnotes_countersaux_prop
88 \bool_new:N \g__postnotes_firststrun_bool
89 \bool_gset_true:N \g__postnotes_firststrun_bool
90 \cs_new_protected:Npn \__postnotes_store_labelseq:nn #1#2
91 {
92   \bool_lazy_any:nT
93   {
94     { \str_if_eq_p:nn {#1} { mark } }
95     { \str_if_eq_p:nn {#1} { section } }
96     { \str_if_eq_p:nn {#1} { preprint } }
97   }
98   { \seq_gput_right:Nn \g__postnotes_labelseq_seq { {#1} {#2} } }
99 }
100 \cs_new_protected:Npn \__postnotes_step_countersaux:nnn #1#2#3
101 {
102   \bool_lazy_and:nnT
103   { \g__postnotes_countersaux_bool }
104   { \str_if_eq_p:nn {#1} { mark } }
105   {
106     \int_gadd:Nn \g__postnotes_postnote_countersaux_int { #3 }
107     \prop_gput:Nne \g__postnotes_countersaux_prop { #2 }
108     { \int_use:N \g__postnotes_postnote_countersaux_int }
109   }
110 }
111 \cs_new_protected:Npe \post@note #1#2#3#4
112 {
113   \exp_not:N \bool_gset_false:N \exp_not:N \g__postnotes_firststrun_bool
114   \exp_not:N \__postnotes_store_labelseq:nn { #1 } { #2 }
115   \exp_not:N \__postnotes_step_countersaux:nnn { #1 } { #2 } { #4 }
116   \exp_not:N \@newl@bel { \c__postnotes_ref_prefix_tl } { #1 @ #2 } { #3 }
117 }

```

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

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

```

118 \AddToHook { begindocument }
119 {
120   \legacy_if:nT { @filesw }
121   {
122     \iow_now:Ne \@mainaux
123     {
124       \token_to_str:N \providecommand
125       \token_to_str:N \post@note [4] { }
126     }
127     \iow_now:Ne \@mainaux
128     {
129       \token_to_str:N \providecommand
130       \token_to_str:N \postnote@setcounteraux [1] { }
131     }
132     \iow_now:Ne \@mainaux

```

```

133         {
134             \token_to_str:N \providecommand
135             \token_to_str:N \postnote@addtocounteraux [1] { }
136         }
137     }
138 }
139 \AddToHook { include/before }
140 {
141     \legacy_if:nT { @filesw }
142     {
143         \iow_now:Ne \@mainaux
144         {
145             \token_to_str:N \providecommand
146             \token_to_str:N \post@note [4] { }
147         }
148         \iow_now:Ne \@mainaux
149         {
150             \token_to_str:N \providecommand
151             \token_to_str:N \postnote@setcounteraux [1] { }
152         }
153         \iow_now:Ne \@mainaux
154         {
155             \token_to_str:N \providecommand
156             \token_to_str:N \postnote@addtocounteraux [1] { }
157         }
158     }
159 }

```

`\__postnotes_set_label:n` 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_label:n
\__postnotes_set_mark_page_label:n
\__postnotes_set_section_page_label:n
\__postnotes_set_text_page_label:n
\__postnotes_set_print_page_label:n
\__postnotes_set_pre_print_label:n
\c__postnotes_page_counter_tl

\__postnotes_set_label:n {<label type>} {<note id>} {<value>}
\__postnotes_set_mark_page_label:n {<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>}

160 \cs_new_protected:Npn \__postnotes_set_label:n #1#2#3#4
161 {
162     \legacy_if:nT { @filesw }
163     {
164         \protected@write \@auxout
165         { \cs_set_eq:NN \c__postnotes_page_counter_tl \scan_stop: }
166         { \token_to_str:N \post@note { #1 } { #2 } { #3 } { #4 } }
167     }
168 }
169 \tl_const:Nn \c__postnotes_page_counter_tl { \int_use:N \c@page }
170 \cs_new_protected:Npn \__postnotes_set_mark_page_label:n #1#2
171 { \__postnotes_set_label:n { mark } { #1 } { \thepage } { #2 } }
172 \cs_generate_variant:Nn \__postnotes_set_mark_page_label:n { ee }
173 \cs_new_protected:Npn \__postnotes_set_section_page_label:n #1
174 { \__postnotes_set_label:n { section } { #1 } { \thepage } { } }
175 \cs_generate_variant:Nn \__postnotes_set_section_page_label:n { e }

```

```

176 \cs_new_protected:Npn \__postnotes_set_text_page_label:n #1
177   { \__postnotes_set_label:nmmn { text } { #1 } { \c__postnotes_page_counter_tl } { } }
178 \cs_generate_variant:Nn \__postnotes_set_text_page_label:n { e }
179 \cs_new_protected:Npn \__postnotes_set_print_page_label:n #1
180   { \__postnotes_set_label:nmmn { print } { #1 } { \c__postnotes_page_counter_tl } { } }
181 \cs_generate_variant:Nn \__postnotes_set_print_page_label:n { e }
182 \cs_new_protected:Npn \__postnotes_set_pre_print_label:n #1
183   { \__postnotes_set_label:nmmn { preprint } { #1 } { } { } }
184 \cs_generate_variant:Nn \__postnotes_set_pre_print_label:n { e }

```

(End of definition for `\__postnotes_set_label:nmmn` 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>}

185 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
186   {
187     \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
188       { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
189       { \tl_clear:N #1 }
190   }
191 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Ne }
192 \cs_new:Npn \__postnotes_extract_pageref:n #1
193   {
194     \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
195       { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
196       { \c_empty_tl }
197   }
198 \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

```

199 \keys_define:nn { postnotes/setup }
200   {
201     heading .cs_set_protected:Np = \pnheading ,
202     heading .value_required:n = true ,
203   }

```

`\pnheading`    Provide default value for `\pnheading`.

```

204 \cs_if_exist:cTF { chapter }
205   {
206     \cs_new_protected:Npn \pnheading
207       {
208         \chapter*{\pntitle}
209         \@mkboth{\pnheaderdefault}{\pnheaderdefault}
210       }
211   }
212 {

```

```

213 \cs_new_protected:Npn \pnheading
214 {
215   \section*{\pntitle}
216   \@mkboth{\pnheaderdefault}{\pnheaderdefault}
217 }
218 }

```

(End of definition for `\pnheading`.)

## format option

```

219 \tl_new:N \l__postnotes_print_format_tl
220 \keys_define:nn { postnotes/setup }
221 {
222   format .tl_set:N = \l__postnotes_print_format_tl ,
223   format .initial:n = { \small } ,
224   format .value_required:n = true ,
225 }

```

## listenv option

```

226 \tl_new:N \l__postnotes_print_env_tl
227 \bool_new:N \l__postnotes_print_as_list_bool
228 \keys_define:nn { postnotes/setup }
229 {
230   listenv .code:n =
231   {
232     \tl_if_eq:nnTF {#1} { none }
233     {
234       \bool_set_false:N \l__postnotes_print_as_list_bool
235       \tl_set:Nn \l__postnotes_post_printnote_tl { \par }

```

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

```

236       \tl_set:Nn \l__postnotes_print_env_tl { itemize }
237     }
238     {
239       \bool_set_true:N \l__postnotes_print_as_list_bool
240       \tl_set:Nn \l__postnotes_print_env_tl {#1}
241     }
242   } ,
243   listenv .initial:n = { postnoteslist } ,
244   listenv .value_required:n = true ,
245 }

```

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<sup>A</sup>T<sub>E</sub>X Companion*).

```

246 \NewDocumentEnvironment { postnoteslist } { }
247 {
248   \list { }
249   {
250     \setlength { \leftmargin } { Opt }
251     \setlength { \labelwidth } { Opt }
252     \setlength { \itemindent } { .5\parindent }
253     \cs_set_eq:NN \makelabel \__postnotes_list_makelabel:n

```



```

254     \setlength { \rightmargin } { Opt }
255     \setlength { \listparindent } { \parindent }
256     \setlength { \parsep } { \parskip }
257     \setlength { \itemsep } { Opt }
258     \setlength { \topsep } { .5\topsep }
259     \setlength { \partopsep } { .5\partopsep }
260 }
261 }
262 { \endlist }
263 \NewDocumentEnvironment { postnoteslisthang } { }
264 {
265   \list { }
266   {
267     \setlength { \leftmargin } { 1em }
268     \setlength { \labelwidth } { -\leftmargin }
269     \setlength { \itemindent } { -2\leftmargin }
270     \cs_set_eq:NN \makelabel \__postnotes_list_makelabel:n
271     \setlength { \rightmargin } { Opt }
272     \setlength { \listparindent } { \parindent }
273     \setlength { \parsep } { \parskip }
274     \setlength { \itemsep } { Opt }
275     \setlength { \topsep } { .5\topsep }
276     \setlength { \partopsep } { .5\partopsep }
277   }
278 }
279 { \endlist }
280 \cs_new:Npn \__postnotes_list_makelabel:n #1
281 { \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.

```

282 \keys_define:nn { postnotes/setup }
283 {
284   makemark .cs_set:Np = \__postnotes_make_mark:nnn #1#2#3 ,
285   makemark .value_required:n = true ,

```

From the default kernel definition of \@makefnmark.

```

286   makemark .initial:n =
287     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
288   maketextmark .cs_set:Np = \__postnotes_make_text_mark:nnn #1#2#3 ,
289   maketextmark .value_required:n = true ,
290   maketextmark .initial:n = { #2 #1 . #3 } ,
291 }
292 \cs_generate_variant:Nn \__postnotes_make_mark:nnn { Vnn }

```

## pretextmark, posttextmark, postprintnote options

```

293 \tl_new:N \l__postnotes_pre_textmark_tl
294 \tl_new:N \l__postnotes_post_textmark_tl
295 \tl_new:N \l__postnotes_post_printnote_tl
296 \keys_define:nn { postnotes/setup }
297 {

```

```

298   pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
299   pretextmark .value_required:n = true ,
300   posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
301   posttextmark .value_required:n = true ,
302   postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
303   postprintnote .value_required:n = true ,
304 }

```

## style option

```

305 \keys_define:nn { postnotes/setup }
306 {
307   style .choice: ,
308   style / endnotes .meta:n =
309   {
310     listenv = none ,
311     format =
312     {
313       \footnotesize
314       \setlength { \rightskip } { Opt }
315       \setlength { \leftskip } { Opt }
316       \setlength { \parindent } { 1.8em }
317     } ,

```

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.

```

318     maketextmark =
319     {
320       \hbox_set:Nn \l__postnotes_tmpa_box
321       { \@textsuperscript { \normalfont ##1 } }
322       \skip_horizontal:n { - \box_wd:N \l__postnotes_tmpa_box }
323       ##2 \box_use:N \l__postnotes_tmpa_box ##3
324     } ,
325   } ,
326   style / pagenote .meta:n =
327   {
328     listenv = none ,
329     format = { } ,
330     pretextmark = { \noindent } ,
331     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
332     posttextmark = { ~ } ,
333   } ,
334 }

```

## hyperref and backlink options

```

335 \bool_new:N \l__postnotes_hyperlink_bool
336 \bool_new:N \l__postnotes_hyperref_warn_bool
337 \bool_new:N \l__postnotes_backlink_bool
338 \keys_define:nn { postnotes/setup }
339 {
340   hyperref .choice: ,
341   hyperref / auto .code:n =
342   {
343     \bool_set_true:N \l__postnotes_hyperlink_bool
344     \bool_set_false:N \l__postnotes_hyperref_warn_bool

```

```

345     } ,
346     hyperref / true .code:n =
347     {
348         \bool_set_true:N \l__postnotes_hyperlink_bool
349         \bool_set_true:N \l__postnotes_hyperref_warn_bool
350     } ,
351     hyperref / false .code:n =
352     {
353         \bool_set_false:N \l__postnotes_hyperlink_bool
354         \bool_set_false:N \l__postnotes_hyperref_warn_bool
355     } ,
356     hyperref .initial:n = auto ,
357     hyperref .default:n = true ,
358     backlink .bool_set:N = \l__postnotes_backlink_bool ,
359     backlink .initial:n = true ,
360     backlink .default:n = true ,
361 }
362 \AddToHook { begindocument }
363 {
364     \IfPackageLoadedTF { hyperref }
365     { }
366     {
367         \bool_if:NT \l__postnotes_hyperref_warn_bool
368         { \msg_warning:nn { postnotes } { missing-hyperref } }
369         \bool_set_false:N \l__postnotes_hyperlink_bool
370     }
371     \keys_define:nn { postnotes/setup }
372     {
373         hyperref .code:n =
374         {
375             \msg_warning:nnn { postnotes }
376             { option-preamble-only } { hyperref }
377         } ,
378         backlink .code:n =
379         {
380             \msg_warning:nnn { postnotes }
381             { option-preamble-only } { backlink }
382         } ,
383     }
384 }
385 \msg_new:nnn { postnotes } { option-preamble-only }
386 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
387 \msg_new:nnn { postnotes } { missing-hyperref }
388 { 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*: `\multiplefootnotemark` and `\multfootsep`. 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 `\multiplefootnotemark`. 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 `\multiplefootnotemark` 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 `\multfootsep` though. `footmisc` stores a plain comma in `\multfootsep` and applies formatting around it in `\FN@mf@check` (hard-coded as `\textsuperscript`). `KOMA-Script` also stores plain content in `\multfootsep` and applies the formatting in the check function. `memoir`, however, stores the formatting directly in `\multfootsep` and applies no formatting later. Also, `latex-lab-footmisc.ltx`, in adding PDF tagging support for `footmisc` stores the tagging code directly in `\multfootsep`. 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 `\multfootsep`, 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 `\multfootsep`. 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`:

```
389 \tl_new:N \l__postnotes_multisep_tl
390 \tl_const:Nn \c_postnotes_multi_notemark_tl { 3sp }
391 \bool_new:N \l__postnotes_multiple_bool
392 \tl_new:N \l__postnotes_saved_spacefactor_multi_tl
393 \keys_define:nn { postnotes/setup }
394 {
395     multiple .bool_set:N = \l__postnotes_multiple_bool ,
396     multiple .default:n = true ,
397     multiple .initial:n = false ,
398     multisep .tl_set:N = \l__postnotes_multisep_tl ,
```

```

399     multisep .value_required:n = true ,
400     multisep .initial:n = {,} ,
401 }

```

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`.

```

402 \cs_new_protected:Npn \__postnotes_multiple_prepare:
403 {
404   \bool_if:NT \l__postnotes_multiple_bool
405   {
406     \kern -\c_postnotes_multi_notemarker_tl
407     \kern \c_postnotes_multi_notemarker_tl
408     \scan_stop:
409   }
410 }

```

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).

```

411 \cs_new_protected:Npn \__postnotes_multiple_check:
412 {
413   \bool_if:NT \l__postnotes_multiple_bool
414   {
415     \dim_compare:nNnT
416     { \c_postnotes_multi_notemarker_tl } = { \lastkern }
417     {
418       \tl_set:Ne \l__postnotes_saved_spacefactor_multi_tl
419       { \int_use:N \spacefactor }
420       \unkern
421       \unkern
422       \tag_socket_use:n { postnotes/multisep/begin }
423       \__postnotes_make_mark:Vnn \l__postnotes_multisep_tl { } { }
424       \tag_socket_use:n { postnotes/multisep/end }
425       \spacefactor \l__postnotes_saved_spacefactor_multi_tl
426       \scan_stop:
427     }
428   }
429 }

```

### sort option

```

430 \bool_new:N \l__postnotes_sort_bool
431 \keys_define:nn { postnotes/setup }
432 {
433   sort .bool_set:N = \l__postnotes_sort_bool ,
434   sort .initial:n = true ,
435   sort .default:n = true ,
436 }

```

### checkduplicates and checkfloats options

```

437 \bool_new:N \l__postnotes_check_dupli_bool
438 \bool_new:N \l__postnotes_check_floats_bool
439 \keys_define:nn { postnotes/setup }

```

```

440 {
441   checkduplicates .bool_set:N = \l__postnotes_check_dupli_bool ,
442   checkduplicates .default:n = true ,
443   checkduplicates .initial:n = true ,
444   checkfloats .bool_set:N = \l__postnotes_check_floats_bool ,
445   checkfloats .default:n = true ,
446   checkfloats .initial:n = false ,
447 }

```

### maybemulti option

```

448 \bool_new:N \l__postnotes_maybe_multi_bool
449 \keys_define:nn { postnotes/setup }
450 {
451   maybemulti .bool_set:N = \l__postnotes_maybe_multi_bool ,
452   maybemulti .default:n = true ,
453   maybemulti .initial:n = false ,
454 }

```

### counteraux option

```

455 \bool_new:N \g__postnotes_counteraux_bool
456 \keys_define:nn { postnotes/setup }
457 {
458   counteraux .bool_gset:N = \g__postnotes_counteraux_bool ,
459   counteraux .default:n = true ,
460   counteraux .initial:n = false ,
461 }
462 \AddToHook { begindocument/before }
463 {
464   \bool_if:NT \g__postnotes_counteraux_bool
465     { \postnotessetup { sort=false } }
466   \keys_define:nn { postnotes/setup }
467     {
468       counteraux .code:n =
469         {
470           \msg_warning:nnn { postnotes }
471             { option-preamble-only } { counteraux }
472         } ,
473     }
474 }

```

```

\pnsetcounteraux
\pnaddtounteraux
\postnote@setcounteraux
\postnote@addtounteraux
475 \cs_new_protected:Npn \postnote@setcounteraux #1
476   { \int_gset:Nn \g__postnotes_postnote_counteraux_int { #1 } }
477 \cs_new_protected:Npn \postnote@addtounteraux #1
478   { \int_gadd:Nn \g__postnotes_postnote_counteraux_int { #1 } }
479 \NewDocumentCommand \pnsetcounteraux { m }
480 {
481   \@bsphack
482   \legacy_if:nT { @filesw }
483   {

```

```

484     \protected@write \@auxout { }
485     { \token_to_str:N \postnote@setcounteraux { #1 } }
486   }
487   \@esphack
488 }
489 \NewDocumentCommand \pnaddtocounteraux { m }
490 {
491   \@bsphack
492   \legacy_if:nT { @filesw }
493   {
494     \protected@write \@auxout { }
495     { \token_to_str:N \postnote@addtocounteraux { #1 } }
496   }
497   \@esphack
498 }

```

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

## `\postnotesetup`

`\postnotesetup` Provide `\postnotesetup`.

```

\postnotesetup{options}

499 \NewDocumentCommand \postnotesetup { m }
500 { \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.

```
501 \newcounter { postnote }
```

```
\g__postnotes_note_id_int \g__postnotes_note_id_int is the internal, unique counter which provides the ID number of each note. It ties “mark” and “text” together, is also the connection between each
\l__postnotes_note_id_tl
\g__postnotes_queue_seq
\l__postnotes_counteraux_step_int
\l__postnotes_mark_typeset_tl
\l__postnotes_note_set_labels_tl
```

note and its data, including the content, which is stored in a property list named according to `\__postnotes_data_name:n` and the ID number. `\l_postnotes_note_id_tl` is a convenience variable storing the counter's value. `\g__postnotes_queue_seq` stores the sequence of notes' IDs to be processed by the next call of `\printpostnotes`.

```

502 \int_new:N \g__postnotes_note_id_int
503 \tl_new:N \l_postnotes_note_id_tl
504 \tl_set:Nn \l_postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
505 \seq_new:N \g__postnotes_queue_seq
506 \int_new:N \l_postnotes_counteraux_step_int
507 \tl_new:N \l__postnotes_mark_typeset_tl
508 \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}
```

```

509 \NewDocumentCommand \postnote { 0 { } +m }
510 { \__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}
```

```

511 \NewHook { postnotes/note/begin }
512 \cs_new_protected:Npn \__postnotes_note:nn #1#2
513 {
514   \group_begin:
515   \keys_set:nn { postnotes/note } {#1}
516   \bool_if:NT \l__postnotes_nomark_bool { \@bsphack }
517   \__postnotes_inhibit_note:F
518   {
519     \int_gincr:N \g__postnotes_note_id_int
520     \tl_if_empty:NTF \l__postnotes_mark_tl
521     {
522       \stepcounter { postnote }
523       \int_set:Nn \l__postnotes_counteraux_step_int { 1 }
524       \bool_if:NT \g__postnotes_counteraux_bool
525       {
526         \exp_args:NNe \prop_gpop:NnNT \g__postnotes_counteraux_prop
527         { \l_postnotes_note_id_tl } \l__postnotes_tmpa_tl
528         { \int_set:Nn \c@postnote { \l__postnotes_tmpa_tl } }
529         \tl_clear:N \l__postnotes_tmpa_tl
530       }
531       \protected@edef \l__postnotes_mark_tl { \thepostnote }
532     }
533     { \int_set:Nn \l__postnotes_counteraux_step_int { 0 } }
534   \UseHook { postnotes/note/begin }

```

```

535     \seq_gput_right:Ne \g__postnotes_queue_seq
536       { \l_postnotes_note_id_tl }
537     \tl_set:Nn \@currentcounter { postnote }
538     \protected@edef \@currentlabel { \p@postnote \l__postnotes_mark_tl }
539     \tl_gset:Ne \@currentHref
540       { postnote. \l_postnotes_note_id_tl .mark }
541     \__postnotes_store:nn { \l_postnotes_note_id_tl } {#2}
542     \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`.

```

543     \bool_lazy_or:nnT
544       { \g__postnotes_counter_aux_bool }
545       { \l__postnotes_maybe_multi_bool }
546     {
547       \bool_lazy_and:nnT
548         { ! \g__postnotes_firstrun_bool }
549         {
550           ! \cs_if_exist_p:c
551             { \c__postnotes_ref_prefix_tl @ mark @ \l_postnotes_note_id_tl }
552           }
553         { \__postnotes_get_label_if_exist:N \l__postnotes_mark_typeset_tl }
554       }
555     \tl_set:Nn \l__postnotes_note_set_labels_tl
556       {
557         \MakeLinkTarget* { postnote. \l_postnotes_note_id_tl .mark }
558         \__postnotes_set_mark_page_label:ee { \l_postnotes_note_id_tl }
559         { \int_use:N \l__postnotes_counter_aux_step_int }
560         \__postnotes_set_user_labels:
561       }
562     \bool_if:NTF \l__postnotes_nomark_bool
563     {
564       \tag_socket_use:n { postnotes/nomark/begin }
565       \l__postnotes_note_set_labels_tl
566       \tag_socket_use:n { postnotes/nomark/end }
567     }
568     {
569       \__postnotes_typeset_mark:eVN
570       { \l_postnotes_note_id_tl } \l__postnotes_mark_typeset_tl
571       \l__postnotes_note_set_labels_tl
572     }
573   }
574   \bool_if:NT \l__postnotes_nomark_bool { \@esphack }
575   \group_end:
576 }

```

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

Options for `\postnote`.

```

577 \tl_new:N \l__postnotes_mark_tl
578 \bool_new:N \l__postnotes_nomark_bool
579 \fp_new:N \l__postnotes_sort_num_fp
580 \str_new:N \l__postnotes_note_label_str
581 \bool_new:N \l__postnotes_manual_sortnum_bool
582 \keys_define:nn { postnotes/note }

```

```

583 {
584   markstr .tl_set:N = \l__postnotes_mark_tl ,
585   markstr .value_required:n = true ,
586   sortnum .code:n =
587     {
588       \fp_set:Nn \l__postnotes_sort_num_fp {#1}
589       \bool_set_true:N \l__postnotes_manual_sortnum_bool
590     } ,
591   sortnum .value_required:n = true ,
592   mark .meta:n =
593     {
594       markstr = {#1} ,
595       sortnum = {#1} ,
596     } ,
597   mark .value_required:n = true ,
598   nomark .bool_set:N = \l__postnotes_nomark_bool ,
599   nomark .default:n = true ,
600   label .str_set:N = \l__postnotes_note_label_str ,
601   label .value_required:n = true ,
602   maybemulti .bool_set:N = \l__postnotes_maybe_multi_bool ,
603   maybemulti .default:n = true ,
604 }

```

`\__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.

```

605 \bool_new:N \l__postnotes_inhibit_note_bool
606 \bool_new:N \l__postnotes_print_plain_mark_bool
607 \bool_new:N \l__postnotes_print_plain_mark_stepcounter_bool
608 \NewHook { postnotes/note/inhibit }
609 \prg_new_protected_conditional:Npnmn \__postnotes_inhibit_note: { F }
610 {
611   \bool_set_false:N \l__postnotes_inhibit_note_bool
612   \bool_set_false:N \l__postnotes_print_plain_mark_bool
613   \bool_set_false:N \l__postnotes_print_plain_mark_stepcounter_bool
614   \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.

```

615   \bool_lazy_all:nT
616   {

```

```

617     { \l__postnotes_inhibit_note_bool }
618     { \l__postnotes_print_plain_mark_bool }
619     { ! \l__postnotes_nomark_bool }
620   }
621   {
622     \tl_if_empty:NTF \l__postnotes_mark_tl
623     {
624       \bool_if:NTF \l__postnotes_print_plain_mark_stepcounter_bool
625       {
626         \stepcounter { postnote }
627         \protected@edef \l__postnotes_mark_typeset_tl { \thepostnote }
628       }
629       {
630         \group_begin:
631         \int_incr:N \c@postnote
632         \protected@edef \l__postnotes_tmpa_tl { \thepostnote }
633         \exp_args:NNNV
634         \group_end:
635         \tl_set:Nn \l__postnotes_mark_typeset_tl \l__postnotes_tmpa_tl
636       }

```

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.

```

637     \__postnotes_get_label_if_exist:N \l__postnotes_mark_typeset_tl
638   }
639   { \tl_set_eq:NN \l__postnotes_mark_typeset_tl \l__postnotes_mark_tl }
640   \group_begin:
641     \socket_assign_plug:nn { tagssupport/postnotes/multisep/begin } { noop }
642     \socket_assign_plug:nn { tagssupport/postnotes/multisep/end } { noop }
643     \__postnotes_typeset_mark_wrapper:nnn
644     { \__postnotes_make_mark:Vnn \l__postnotes_mark_typeset_tl { } { } }
645     { } { }
646   \group_end:
647   }
648   \bool_if:NTF \l__postnotes_inhibit_note_bool
649   { \prg_return_true: }
650   { \prg_return_false: }
651 }

```

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

```

\__postnotes_get_label_if_exist:N     \__postnotes_get_label_if_exist:N <\l__postnotes_mark_tl>
652 \cs_new_protected:Npn \__postnotes_get_label_if_exist:N #1
653 {
654   \str_if_empty:NTF \l__postnotes_note_label_str
655   {

```

```

656 \str_if_empty:NF \l__postnotes_note_zlabel_str
657 {
658   \exp_args:NV \zref@ifrefundefined \l__postnotes_note_zlabel_str
659   { }
660   {
661     \exp_args:NV \zref@ifrefcontainsprop
662     \l__postnotes_note_zlabel_str
663     { postnote@mark }
664     {
665       \exp_args:NNNo \exp_args:NNNo \tl_set:Nn #1
666       {
667         \zref@extract
668         { \l__postnotes_note_zlabel_str } { postnote@mark }
669       }
670     }
671   }
672 }
673 }
674 }
675 {
676 \exp_args:Ne \property_if_recorded:nnT
677 { postnotes@ \l__postnotes_note_label_str }
678 { postnotes/mark }
679 {
680 \protected@edef #1
681 {
682 \property_ref:ee
683 { __postnotes_ \l__postnotes_note_label_str } { postnotes/mark }
684 }
685 }
686 }
687 }

```

(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>}
688 \cs_new_protected:Npn \__postnotes_typeset_mark:nnN #1#2#3
689 {
690 \__postnotes_typeset_mark_wrapper:nnn
691 {
692 #3
693 \bool_if:NTF \l__postnotes_hyperlink_bool
694 {
695 \__postnotes_make_mark:nnn {#2}
696 { \hyper@linkstart { link } { postnote. #1 .text } }
697 { \hyper@linkend }
698 }
699 { \__postnotes_make_mark:nnn {#2} { } { } }
700 }

```

```

701     { \tag_socket_use:n { postnotes/mark/begin } }
702     { \tag_socket_use:n { postnotes/mark/end }   }
703   }
704 \cs_generate_variant:Nn \l__postnotes_typeset_mark:nnN { eVN }
705 \tl_new:N \l__postnotes_saved_spacefactor_tl
706 \cs_new_protected:Npn \l__postnotes_typeset_mark_wrapper:nnn #1#2#3
707 {
708   \mode_leave_vertical:
709   \mode_if_horizontal:T
710   {
711     \tl_set:Ne \l__postnotes_saved_spacefactor_tl
712     { \int_use:N \spacefactor }
713     \__postnotes_multiple_check:
714     \nobreak
715   }
716   #2 % begin tagging socket
717   #1 % mark
718   #3 % end tagging socket
719   \__postnotes_multiple_prepare:
720   \mode_if_horizontal:T
721   { \spacefactor \l__postnotes_saved_spacefactor_tl }
722   \scan_stop:
723 }

```

*(End of definition for \l\_\_postnotes\_typeset\_mark:nnN and \l\_\_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`.

```

724 \cs_new_protected:Npn \__postnotes_set_user_labels:
725 {
726   \str_if_empty:NF \l__postnotes_note_label_str
727   {
728     \exp_args:NV \label \l__postnotes_note_label_str
729     \property_record:ee { __postnotes_ \l__postnotes_note_label_str }
730     { postnotes/mark }
731   }
732   \str_if_empty:NF \l__postnotes_note_zlabel_str
733   { \exp_args:NV \zlabel \l__postnotes_note_zlabel_str }
734 }
735 \property_new:nmmm { postnotes/mark } { now } { } { \l__postnotes_mark_tl }

```

*(End of definition for \\_\_postnotes\_set\_user\_labels:.)*

## 5 \postnoteref

`\postnoteref` Provide `\postnoteref`.

```

\postnoteref(*){\label}
736 \NewDocumentCommand \postnoteref { s m }
737 { \__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}}
738 \str_new:N \l__postnotes_note_ref_label_str
739 \cs_new_protected:Npn \__postnotes_note_ref:nn #1#2
740 {
741   \group_begin:
742     \str_set:Nn \l__postnotes_note_ref_label_str {#2}
743     \__postnotes_typeset_mark_wrapper:nnn
744     {
745       \bool_lazy_and:nnTF
746       { ! #1 }
747       { \l__postnotes_hyperlink_bool }
748       {
749         \hyperref [#2]
750         { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
751       }
752       { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
753     }
754     { \tag_socket_use:n { postnotes/postnoteref/begin } }
755     { \tag_socket_use:n { postnotes/postnoteref/end } }
756   \group_end:
757 }
```

*(End of definition for `\__postnotes_note_ref:nn`.)*

## 6 `\postnotesection`

`\postnotesection` Provide `\postnotesection`.

```

    \postnotesection[<options>]{<section content>}
758 \NewDocumentCommand \postnotesection { 0 { } +m }
759 {
760   \@bsphack
761   \__postnotes_section:nn {#1} {#2}
762   \@esphack
763 }
```

*(End of definition for `\postnotesection`.)*

`\__postnotes_section:nn` The internal version of `\postnotesection`.

```

    \__postnotes_section:nn {\options} {\content}}
764 \int_new:N \g__postnotes_sectid_int
765 \cs_new_protected:Npn \__postnotes_section:nn #1#2
766 {
767   \group_begin:
768     \int_gincr:N \g__postnotes_sectid_int
769     \int_gincr:N \g__postnotes_note_id_int
770     \seq_gput_right:Ne \g__postnotes_queue_seq { \l__postnotes_note_id_tl }
771     \tl_gclear:N \g__postnotes_section_name_tl
```



```

772     \keys_set:nn { postnotes/section } {#1}
773     \__postnotes_set_section_page_label:e { \l_postnotes_note_id_tl }
774     \bool_if:NTF \l__postnotes_section_exp_bool
775     {
776         \protected@edef \l__postnotes_tmpa_tl {#2}
777         \__postnotes_store_section:nV { \l_postnotes_note_id_tl }
778         \l__postnotes_tmpa_tl
779     }
780     { \__postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
781 \group_end:
782 }

```

(End of definition for `\__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.

```

783 \tl_new:N \g__postnotes_section_name_tl
784 \bool_new:N \l__postnotes_section_exp_bool
785 \keys_define:nn { postnotes/section }
786 {
787     name .tl_gset:N = \g__postnotes_section_name_tl ,
788     name .value_required:n = true ,
789     exp .bool_set:N = \l__postnotes_section_exp_bool ,
790     exp .initial:n = false ,
791     exp .default:n = true ,
792 }

```

## 7 `\printpostnotes`

`\printpostnotes` Provide `\printpostnotes`.

```

\printpostnotes

793 \NewDocumentCommand \printpostnotes { }
794 { \__postnotes_print_notes: }

```

(End of definition for `\printpostnotes`.)

<code>\pnthechapter</code>	User facing variables, aimed at making available some of the notes’ and sections’ metadata
<code>\pnthesection</code>	for the user at specific contexts.
<code>\pnthechapternextnote</code>	795 <code>\tl_new:N \pnthechapter</code>
<code>\pnthesectionnextnote</code>	796 <code>\tl_new:N \pnthesection</code>
<code>\pnthepage</code>	797 <code>\tl_new:N \pnidnextnote</code>
<code>\pnidnextnote</code>	798 <code>\tl_new:N \pnthechapternextnote</code>
	799 <code>\tl_new:N \pnthesectionnextnote</code>
	800 <code>\tl_new:N \pnthepage</code>

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

```

\g_postnotes_print_postnotes_int Auxiliary variables for \__postnotes_print_notes:.
  \g_postnotes_print_queue_seq 801 \int_new:N \g_postnotes_print_postnotes_int
  \l_postnotes_print_note_id_tl 802 \seq_new:N \g_postnotes_print_queue_seq
\l__postnotes_print_note_id_next_tl 803 \tl_new:N \l_postnotes_print_note_id_tl
  \l__postnotes_print_counter_tl 804 \tl_new:N \l__postnotes_print_note_id_next_tl
\l__postnotes_print_mark_tl 805 \tl_new:N \l__postnotes_print_counter_tl
  \l__postnotes_print_typeset_mark_tl 806 \tl_new:N \l__postnotes_print_mark_tl
  \l__postnotes_print_type_curr_tl 807 \tl_new:N \l__postnotes_print_typeset_mark_tl
  \l__postnotes_print_type_next_tl 808 \tl_new:N \l__postnotes_print_type_curr_tl
  \l__postnotes_print_type_prev_tl 809 \tl_new:N \l__postnotes_print_type_next_tl
  \l__postnotes_print_content_tl 810 \tl_new:N \l__postnotes_print_type_prev_tl
  \l__postnotes_clear_queue_seq 811 \tl_new:N \l__postnotes_print_content_tl
  812 \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.

```

813 \NewHook { postnotes/print/begin }
814 \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.

```

815 \newcounter { postnotetext }
816 \newcounter { postnotesection }
817 \setcounter { postnotesection } { 10000 }

```

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

```

\__postnotes_print_notes:

```

```

818 \cs_new_protected:Npn \__postnotes_print_notes:
819 {
820   \group_begin:
821   \int_gincr:N \g__postnotes_print_postnotes_int
822   \__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`.

```

823   \bool_lazy_and:nnTF
824     { \g__postnotes_countersaux_bool }
825     { ! \g__postnotes_firstrun_bool }
826     {
827       \seq_gset_eq:NN \g__postnotes_print_queue_seq
828       \g__postnotes_print_labelseq_queue_seq
829     }
830     {
831       \seq_gset_eq:NN \g__postnotes_print_queue_seq
832       \g__postnotes_queue_seq
833     }
834   \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_print_queue_seq
835   \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.

```

836   \__postnotes_set_pre_print_label:e
837     { \int_use:N \g__postnotes_print_postnotes_int }
838   \seq_if_empty:NTF \g__postnotes_print_queue_seq
839     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
840     {
841       \pnheading
842       \UseHook { postnotes/print/begin }
843       \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
844       \__postnotes_check_duplicates:N \g__postnotes_print_queue_seq
845       \__postnotes_sort_queue:N \g__postnotes_print_queue_seq
846       \__postnotes_check_floats:N \g__postnotes_print_queue_seq
847       \bool_gset_true:N \g__postnotes_header_vars_next_bool
848       \__postnotes_get_headers_data:N \g__postnotes_print_queue_seq
849       \__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 `\enoteheading`, 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](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).

```

850   \bool_if:NF \l__postnotes_print_as_list_bool
851     {
852       \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
853       \@afterindenttrue
854     }

```

```

855 \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_print_queue_seq }
856 {
857   \seq_gpop_left:NN \g__postnotes_print_queue_seq
858   \l_postnotes_print_note_id_tl
859   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
860   { type } \l__postnotes_print_type_curr_tl
861   \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
862   { % type_curr = 'section'
863     \seq_if_empty:NTF \g__postnotes_print_queue_seq
864     {
865       \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
866       \tl_set:Nn \l__postnotes_print_type_next_tl { close }
867     }
868     {
869       \seq_get_left:NN \g__postnotes_print_queue_seq
870       \l__postnotes_print_note_id_next_tl
871       \__postnotes_prop_get:nnN
872       { \l__postnotes_print_note_id_next_tl }
873       { type } \l__postnotes_print_type_next_tl
874     }

```

We only process the entry if type\_next is note: here are skipped empty sections.

```

875   \tl_if_eq:NnT \l__postnotes_print_type_next_tl { note }
876   {
877     \stepcounter { postnotessection }
878     \group_begin:
879     \__postnotes_prop_get:nnN
880     { \l_postnotes_print_note_id_tl }
881     { thechapter } \pnthechapter
882     \__postnotes_prop_get:nnN
883     { \l_postnotes_print_note_id_tl }
884     { thesection } \pnthesection
885     \tl_set:NV \pnidnextnote
886     \l__postnotes_print_note_id_next_tl
887     \__postnotes_prop_get:nnN
888     { \l__postnotes_print_note_id_next_tl }
889     { thechapter } \pnthechapternextnote
890     \__postnotes_prop_get:nnN
891     { \l__postnotes_print_note_id_next_tl }
892     { thesection } \pnthesectionnextnote
893     \__postnotes_prop_get:nnN
894     { \l_postnotes_print_note_id_tl }
895     { content } \l__postnotes_print_content_tl
896     \l__postnotes_print_content_tl
897     \group_end:

```

Set type\_prev for the next iteration.

```

898     \tl_set:NV \l__postnotes_print_type_prev_tl
899     \l__postnotes_print_type_curr_tl
900   }
901 }
902 { % type_curr = 'note'
903   \seq_if_empty:NTF \g__postnotes_print_queue_seq
904   {
905     \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }

```

```

906     \tl_set:Nn \l__postnotes_print_type_next_tl { close }
907   }
908   {
909     \seq_get_left:NN \g__postnotes_print_queue_seq
910     \l__postnotes_print_note_id_next_tl
911     \__postnotes_prop_get:nnN
912     { \l__postnotes_print_note_id_next_tl }
913     { type } \l__postnotes_print_type_next_tl
914   }
915 \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }
916 {
917   \bool_if:NTF \l__postnotes_print_as_list_bool
918   { \exp_args:NV \begin \l__postnotes_print_env_tl }
919   { \group_begin: }
920   \tag_socket_use:n { postnotes/printlist/begin }
921   \l__postnotes_print_format_tl
922 }
923 \group_begin:
924 \UseHook { postnotes/print/note/begin }
925 \__postnotes_get_pageref:Ne \pnthepage
926 { mark@ \l__postnotes_print_note_id_tl }
927 \__postnotes_prop_get:nnN
928 { \l__postnotes_print_note_id_tl }
929 { mark } \l__postnotes_print_mark_tl
930 \__postnotes_prop_get:nnN
931 { \l__postnotes_print_note_id_tl }
932 { counter } \l__postnotes_print_counter_tl
933 \__postnotes_prop_get:nnN
934 { \l__postnotes_print_note_id_tl }
935 { content } \l__postnotes_print_content_tl
936 \tl_set:Nn \@currentcounter { postnotetext }
937 \int_set:Nn \c@postnotetext
938 { \l__postnotes_print_counter_tl }
939 \protected@edef \@currentlabel
940 { \p@postnote \l__postnotes_print_mark_tl }
941 \tl_set:Nn \l__postnotes_print_typeset_mark_tl
942 {
943   \tag_socket_use:n { postnotes/printmark/begin }
944   \MakeLinkTarget*
945   { postnote. \l__postnotes_print_note_id_tl .text }
946   \__postnotes_set_text_page_label:e
947   { \l__postnotes_print_note_id_tl }
948   \l__postnotes_pre_textmark_tl
949   \__postnotes_typeset_text_mark:eV
950   { \l__postnotes_print_note_id_tl }
951   \l__postnotes_print_mark_tl
952   \l__postnotes_post_textmark_tl
953   \tag_socket_use:n { postnotes/printmark/end }
954 }

```

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).

```

955         \bool_if:NTF \l__postnotes_print_as_list_bool
956         {
957             \item
958             [
959                 \tag_socket_use:n { postnotes/printnote/begin }
960                 \l__postnotes_print_typeset_mark_tl
961             ]
962             \mode_leave_vertical:
963         }
964         {
965             \mode_leave_vertical:
966             \tag_socket_use:n { postnotes/printnote/begin }
967             \l__postnotes_print_typeset_mark_tl
968         }
969         \tag_socket_use:n { postnotes/printtext/begin }
970         \l__postnotes_print_content_tl
971         \tag_socket_use:n { postnotes/printtext/end }
972         \tag_socket_use:n { postnotes/printnote/end }
973         \l__postnotes_post_printnote_tl
974     \group_end:
975     \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
976     {
977         \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).

```

978         \bool_if:NTF \l__postnotes_print_as_list_bool
979         {
980             \exp_args:NV \end \l__postnotes_print_env_tl
981             \par
982         }
983         {
984             \par
985             \group_end:
986         }
987     }

```

Set type\_prev for the next iteration.

```

988         \tl_set:NV \l__postnotes_print_type_prev_tl
989         \l__postnotes_print_type_curr_tl
990     }
991 }
992 \AddToHookNext { shipout/after }
993 { \bool_gset_false:N \g__postnotes_header_vars_next_bool }

```

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

```

994     \seq_map_inline:Nn \l__postnotes_clear_queue_seq
995     { \__postnotes_prop_gc_clear:n { ##1 } }
996 }
997 \group_end:
998 }

```

(End of definition for `\__postnotes_print_notes:`)

```
999 \msg_new:nnn { postnotes } { empty-printpostnotes }
1000 { 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>}
1001 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
1002 {
1003   \bool_lazy_and:nnTF
1004     { \l__postnotes_hyperlink_bool }
1005     { \l__postnotes_backlink_bool }
1006     {
1007       \__postnotes_make_text_mark:nnn {#2}
1008       { \hyper@linkstart { link } { postnote. #1 .mark } }
1009       { \hyper@linkend }
1010     }
1011     { \__postnotes_make_text_mark:nnn {#2} { } { } }
1012 }
1013 \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:`

```
1014 \seq_new:N \g__postnotes_print_labelseq_queue_seq
1015 \cs_new_protected:Npn \__postnotes_split_labelseq:
1016 {
1017   \group_begin:
1018     \seq_clear:N \l__postnotes_tmpa_seq
1019     \seq_clear:N \l__postnotes_tmpb_seq
1020     \bool_if:NTF \g__postnotes_countersaux_bool
1021     {
1022       \tl_set:Nc \l__postnotes_tmpa_tl
1023         { { 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.

```

1024     \seq_if_in:NVF \g__postnotes_labelseq_seq \l__postnotes_tmpa_tl
1025     { \seq_gput_right:NV \g__postnotes_labelseq_seq \l__postnotes_tmpa_tl }
1026     \bool_do_until:nn
1027     { \tl_if_eq_p:NN \l__postnotes_tmpb_tl \l__postnotes_tmpa_tl }
1028     {
1029         \seq_gpop_left:NN \g__postnotes_labelseq_seq \l__postnotes_tmpb_tl
1030         \str_case:enT
1031         { \tl_item:Nn \l__postnotes_tmpb_tl { 1 } }
1032         {
1033             { mark } { }
1034             { section } { }
1035         }
1036     }

```

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.

```

1037         \int_compare:nNnTF
1038         { \tl_item:Nn \l__postnotes_tmpb_tl { 2 } }
1039         >
1040         { \g__postnotes_note_id_int }
1041         {
1042             \seq_put_right:Ne \l__postnotes_tmpb_seq
1043             \l__postnotes_tmpb_tl
1044         }
1045         {
1046             \seq_put_right:Ne \l__postnotes_tmpa_seq
1047             { \tl_item:Nn \l__postnotes_tmpb_tl { 2 } }
1048         }
1049     }

```

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.

```

1050     }
1051     \seq_gset_eq:NN \g__postnotes_print_labelseq_queue_seq
1052     \l__postnotes_tmpa_seq
1053     \seq_concat:NNN \l__postnotes_tmpa_seq \l__postnotes_tmpb_seq
1054     \g__postnotes_labelseq_seq
1055     \seq_gset_eq:NN \g__postnotes_labelseq_seq \l__postnotes_tmpa_seq
1056 }
1057 {

```



```

1058     \seq_map_inline:Nn \g__postnotes_labelseq_seq
1059     {
1060         \str_case:enT
1061         { \tl_item:nn { ##1 } { 1 } }
1062         {
1063             { mark } { }
1064             { section } { }
1065         }
1066         {
1067             \int_compare:nNnTF
1068             { \tl_item:nn { ##1 } { 2 } }
1069             >
1070             { \g__postnotes_note_id_int }
1071             { \seq_put_right:Nn \l__postnotes_tmpb_seq { ##1 } }
1072             {
1073                 \seq_put_right:Ne \l__postnotes_tmpa_seq
1074                 { \tl_item:nn { ##1 } { 2 } }
1075             }
1076         }

```

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.

```

1077     }
1078     \seq_gset_eq:NN \g__postnotes_print_labelseq_queue_seq
1079     \l__postnotes_tmpa_seq
1080     \seq_gset_eq:NN \g__postnotes_labelseq_seq \l__postnotes_tmpb_seq
1081 }
1082 \group_end:
1083 }

```

*(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)
1084 \cs_new_protected:Npn \__postnotes_check_duplicates:N #1
1085 {

```

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

```

1086     \bool_lazy_and:nnT
1087     { ! \g__postnotes_firstrun_bool }

```

```

1088     { ! \g__postnotes_countersaux_bool }
1089     {
1090     \group_begin:
1091     \seq_clear:N \l__postnotes_tmpa_seq
1092     \seq_map_inline:Nn #1
1093     {
1094     \bool_lazy_or:nnTF
1095     { \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.

```

1096     {
1097     \str_if_eq_p:ee
1098     { \__postnotes_prop_item:nn {##1} { type } } { section }
1099     }
1100     { \seq_put_right:Nn \l__postnotes_tmpa_seq {##1} }
1101     {

```

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

```

1102     \str_if_eq:eeF
1103     { \__postnotes_prop_item:nn {##1} { multibool } }
1104     { true }
1105     {
1106     \seq_put_right:Nn \l__postnotes_tmpa_seq {##1}
1107     \bool_if:NT \l__postnotes_check_dupli_bool
1108     {
1109     \msg_warning:nne { postnotes } { possible-duplicate }
1110     { \__postnotes_prop_item:nn {##1} { mark } }
1111     }
1112     }
1113     }
1114     }
1115     \seq_gset_eq:NN #1 \l__postnotes_tmpa_seq
1116     \group_end:
1117     }
1118     }

```

*(End of definition for \\_\_postnotes\_check\_duplicates:N.)*

```

1119 \msg_new:nnn { postnotes } { possible-duplicate }
1120 { Possible~duplicate~*around*~note~'#1'~\msg_line_context:. }

```

```

\__postnotes_check_floats:N     \__postnotes_check_floats:N <\g__postnotes_print_queue_seq>
1121 \cs_new_protected:Npn \__postnotes_check_floats:N #1
1122 {
1123 \bool_lazy_and:nnT
1124 { \l__postnotes_check_floats_bool }

```

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

```

1125 { ! \g__postnotes_firststrun_bool }
1126 {
1127 \group_begin:

```

Only compare sequence net of non-existing labels.

```

1128     \seq_set_filter:NNn \l__postnotes_tmpa_seq #1
1129     {
1130         \bool_lazy_or_p:nn
1131         { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ mark @ ##1 } }
1132         { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ section @ ##1 } }
1133     }

```

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.

```

1134     \tl_if_eq:NNF
1135     \g__postnotes_print_labelseq_queue_seq \l__postnotes_tmpa_seq
1136     { \msg_warning:nn { postnotes } { possible-shuffle } }
1137 \group_end:
1138 }
1139 }

```

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

```

1140 \msg_new:nnn { postnotes } { possible-shuffle }
1141 { 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)

1142 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
1143 {
1144     \bool_if:NT \l__postnotes_sort_bool
1145     {
1146         \group_begin:
1147         \seq_gsort:Nn #1
1148         {
1149             \__postnotes_prop_get:nnN {##1} { pnsectid } \l__postnotes_tmpa_tl
1150             \__postnotes_prop_get:nnN {##2} { pnsectid } \l__postnotes_tmpb_tl
1151             \tl_if_eq:NNTF \l__postnotes_tmpa_tl \l__postnotes_tmpb_tl
1152             {
1153                 \__postnotes_prop_get:nnN {##1} { type } \l__postnotes_tmpa_tl
1154                 \__postnotes_prop_get:nnN {##2} { type } \l__postnotes_tmpb_tl
1155                 \bool_lazy_and:nnTF
1156                 { \str_if_eq_p:Vn \l__postnotes_tmpa_tl { note } }
1157                 { \str_if_eq_p:Vn \l__postnotes_tmpb_tl { note } }
1158                 {
1159                     \__postnotes_prop_get:nnN {##1}
1160                     { sortnum } \l__postnotes_tmpa_tl
1161                     \__postnotes_prop_get:nnN {##2}
1162                     { sortnum } \l__postnotes_tmpb_tl
1163                     \fp_compare:nNnTF
1164                     { \l__postnotes_tmpa_tl } > { \l__postnotes_tmpb_tl }
1165                     { \sort_return_swapped: }
1166                     { \sort_return_same: }
1167                 }
1168                 { \sort_return_same: }
1169             }
1170             { \sort_return_same: }

```

```

1171     }
1172     \group_end:
1173 }
1174 }

(End of definition for \__postnotes_sort_queue:N.)

1175 \AddToHook { enddocument/afterlastpage }
1176 {
1177     \group_begin:
1178     \bool_if:NTF \g__postnotes_counteraux_bool
1179     {
1180         \seq_set_filter:NNn \l__postnotes_tmpa_seq \g__postnotes_labelseq_seq
1181         { \str_if_eq_p:ee { \tl_item:nm {#1} { 1 } } { mark } }
1182     }
1183     {
1184         \seq_set_filter:NNn \l__postnotes_tmpa_seq \g__postnotes_queue_seq
1185         { \str_if_eq_p:ee { \__postnotes_prop_item:nm {#1} { type } } { note } }
1186     }
1187     \seq_if_empty:NF \l__postnotes_tmpa_seq
1188     {
1189         \msg_warning:nne { postnotes } { stray-notes }
1190         { \seq_count:N \l__postnotes_tmpa_seq }
1191     }
1192     \group_end:
1193 }
1194 \msg_new:nnn { postnotes } { stray-notes }
1195 {
1196     There-are~'#1'~stray~notes~after~the~last~'\iow_char:N\printpostnotes'~
1197     \msg_line_context:..At~this~point,~they~are~lost.
1198 }

```

## 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:n`, `\__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).

3. 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>	1199 <code>\tl_new:N \pnhdpagefirst</code>
<code>\pnhdsectfirst</code>	1200 <code>\tl_new:N \pnhdpagelast</code>
<code>\pnhdsectlast</code>	1201 <code>\tl_new:N \pnhdchapfirst</code>
<code>\pnhdnamefirst</code>	1202 <code>\tl_new:N \pnhdchapl原因</code>
<code>\pnhdnamelast</code>	1203 <code>\tl_new:N \pnhdsectfirst</code>
	1204 <code>\tl_new:N \pnhdsectlast</code>
	1205 <code>\tl_new:N \pnhdnamefirst</code>
	1206 <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>	1207 <code>\prop_new:N \g__postnotes_header_page_first_prop</code>
<code>\g__postnotes_header_chap_first_prop</code>	1208 <code>\prop_new:N \g__postnotes_header_page_last_prop</code>
<code>\g__postnotes_header_chap_last_prop</code>	1209 <code>\prop_new:N \g__postnotes_header_chap_first_prop</code>
<code>\g__postnotes_header_sect_first_prop</code>	1210 <code>\prop_new:N \g__postnotes_header_chap_last_prop</code>
<code>\g__postnotes_header_sect_last_prop</code>	1211 <code>\prop_new:N \g__postnotes_header_sect_first_prop</code>
<code>\g__postnotes_header_name_first_prop</code>	1212 <code>\prop_new:N \g__postnotes_header_sect_last_prop</code>
<code>\g__postnotes_header_name_last_prop</code>	1213 <code>\prop_new:N \g__postnotes_header_name_first_prop</code>
<code>\g__postnotes_header_prev_last_page_tl</code>	1214 <code>\prop_new:N \g__postnotes_header_name_last_prop</code>
<code>\g__postnotes_header_prev_last_chap_tl</code>	1215 <code>\tl_new:N \g__postnotes_header_prev_last_page_tl</code>
<code>\g__postnotes_header_prev_last_sect_tl</code>	1216 <code>\tl_new:N \g__postnotes_header_prev_last_chap_tl</code>
<code>\g__postnotes_header_prev_last_name_tl</code>	1217 <code>\tl_new:N \g__postnotes_header_prev_last_sect_tl</code>
<code>\l__postnotes_prev_text_page_tl</code>	1218 <code>\tl_new:N \g__postnotes_header_prev_last_name_tl</code>
<code>\l__postnotes_curr_text_page_tl</code>	1219 <code>\tl_new:N \l__postnotes_prev_text_page_tl</code>
<code>\l__postnotes_prev_mark_page_tl</code>	1220 <code>\tl_new:N \l__postnotes_curr_text_page_tl</code>
<code>\l__postnotes_prev_mark_chap_tl</code>	1221 <code>\tl_new:N \l__postnotes_prev_mark_page_tl</code>
<code>\l__postnotes_prev_mark_sect_tl</code>	1222 <code>\tl_new:N \l__postnotes_prev_mark_chap_tl</code>
<code>\l__postnotes_prev_mark_name_tl</code>	1223 <code>\tl_new:N \l__postnotes_prev_mark_sect_tl</code>
	1224 <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)
1225 \cs_new_protected:Npn \__postnotes_get_headers_data:N #1
1226 {
1227   \group_begin:
1228     \tl_gclear:N \pnhdpagefirst
1229     \tl_gclear:N \pnhdpagelast
1230     \tl_gclear:N \pnhdchapfirst
1231     \tl_gclear:N \pnhdchaplast
1232     \tl_gclear:N \pnhdsectfirst
1233     \tl_gclear:N \pnhdsectlast
1234     \tl_gclear:N \pnhdnamefirst
1235     \tl_gclear:N \pnhdnamelast
1236     \prop_gclear:N \g__postnotes_header_page_first_prop
1237     \prop_gclear:N \g__postnotes_header_page_last_prop
1238     \prop_gclear:N \g__postnotes_header_chap_first_prop
1239     \prop_gclear:N \g__postnotes_header_chap_last_prop
1240     \prop_gclear:N \g__postnotes_header_sect_first_prop
1241     \prop_gclear:N \g__postnotes_header_sect_last_prop
1242     \prop_gclear:N \g__postnotes_header_name_first_prop
1243     \prop_gclear:N \g__postnotes_header_name_last_prop
1244     \tl_gclear:N \g__postnotes_header_prev_last_page_tl
1245     \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
1246     \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
1247     \tl_gclear:N \g__postnotes_header_prev_last_name_tl
1248     \tl_clear:N \l__postnotes_prev_text_page_tl
1249     \tl_clear:N \l__postnotes_curr_text_page_tl
1250     \tl_clear:N \l__postnotes_prev_mark_page_tl
1251     \tl_clear:N \l__postnotes_prev_mark_chap_tl
1252     \tl_clear:N \l__postnotes_prev_mark_sect_tl
1253     \tl_clear:N \l__postnotes_prev_mark_name_tl
1254     \seq_map_inline:Nn #1
1255     {
1256       \exp_args:Ne \tl_if_eq:nnT
1257       { \__postnotes_prop_item:nn {##1} { type } }
1258       { note }
1259       {
1260         \__postnotes_get_pageref:Nn
1261         \l__postnotes_curr_text_page_tl { text@ ##1 }
1262         \tl_if_empty:NF \l__postnotes_curr_text_page_tl
1263         {
1264           \tl_if_eq:NNTF
1265           \l__postnotes_prev_text_page_tl
1266           \l__postnotes_curr_text_page_tl
1267           {
1268             \__postnotes_get_pageref:Nn
1269             \l__postnotes_prev_mark_page_tl { mark@ ##1 }
1270             \__postnotes_prop_get:nnN {##1} { thechapter }

```

```

1271         \l__postnotes_prev_mark_chap_tl
1272         \__postnotes_prop_get:nnN {##1} { thesection }
1273         \l__postnotes_prev_mark_sect_tl
1274         \__postnotes_prop_get:nnN {##1} { pnsectname }
1275         \l__postnotes_prev_mark_name_tl
1276     }
1277 {

```

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).

```

1278         \tl_if_empty:NF \l__postnotes_prev_text_page_tl
1279         {
1280             \prop_gput:NVV \g__postnotes_header_page_last_prop
1281             \l__postnotes_prev_text_page_tl
1282             \l__postnotes_prev_mark_page_tl
1283             \prop_gput:NVV \g__postnotes_header_chap_last_prop
1284             \l__postnotes_prev_text_page_tl
1285             \l__postnotes_prev_mark_chap_tl
1286             \prop_gput:NVV \g__postnotes_header_sect_last_prop
1287             \l__postnotes_prev_text_page_tl
1288             \l__postnotes_prev_mark_sect_tl
1289             \prop_gput:NVV \g__postnotes_header_name_last_prop
1290             \l__postnotes_prev_text_page_tl
1291             \l__postnotes_prev_mark_name_tl
1292         }

```

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

```

1293         \prop_gput:NVe \g__postnotes_header_page_first_prop
1294         \l__postnotes_curr_text_page_tl
1295         { \__postnotes_extract_pageref:n { mark@ ##1 } }
1296         \__postnotes_prop_get:nnN {##1} { thechapter }
1297         \l__postnotes_tmpa_tl
1298         \prop_gput:NVV \g__postnotes_header_chap_first_prop
1299         \l__postnotes_curr_text_page_tl
1300         \l__postnotes_tmpa_tl
1301         \__postnotes_prop_get:nnN {##1} { thesection }
1302         \l__postnotes_tmpa_tl
1303         \prop_gput:NVV \g__postnotes_header_sect_first_prop
1304         \l__postnotes_curr_text_page_tl
1305         \l__postnotes_tmpa_tl
1306         \__postnotes_prop_get:nnN {##1} { pnsectname }
1307         \l__postnotes_tmpa_tl
1308         \prop_gput:NVV \g__postnotes_header_name_first_prop
1309         \l__postnotes_curr_text_page_tl
1310         \l__postnotes_tmpa_tl

```

Store prev\_mark data for the first note on the page.

```

1311         \__postnotes_get_pageref:Nn
1312         \l__postnotes_prev_mark_page_tl { mark@ ##1 }
1313         \__postnotes_prop_get:nnN {##1} { thechapter }

```

```

1314         \l__postnotes_prev_mark_chap_tl
1315         \__postnotes_prop_get:nnN {##1} { thesection }
1316         \l__postnotes_prev_mark_sect_tl
1317         \__postnotes_prop_get:nnN {##1} { pnsectname }
1318         \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).

1319         \tl_set:NV \l__postnotes_prev_text_page_tl
1320         \l__postnotes_curr_text_page_tl
1321     }
1322 }
1323 }
1324 }

```

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.

```

1325     \tl_if_empty:NF \l__postnotes_prev_text_page_tl
1326     {
1327         \prop_gput:NVV \g__postnotes_header_page_last_prop
1328         \l__postnotes_prev_text_page_tl
1329         \l__postnotes_prev_mark_page_tl
1330         \prop_gput:NVV \g__postnotes_header_chap_last_prop
1331         \l__postnotes_prev_text_page_tl
1332         \l__postnotes_prev_mark_chap_tl
1333         \prop_gput:NVV \g__postnotes_header_sect_last_prop
1334         \l__postnotes_prev_text_page_tl
1335         \l__postnotes_prev_mark_sect_tl
1336         \prop_gput:NVV \g__postnotes_header_name_last_prop
1337         \l__postnotes_prev_text_page_tl
1338         \l__postnotes_prev_mark_name_tl
1339     }
1340     \group_end:
1341 }

```

*(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)}

1342     \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
1343     {

```



```

1344 \group_begin:
1345 \prop_get:NnNTF \g__postnotes_header_page_first_prop
1346   {#1} \l__postnotes_tmpa_tl
1347   { \tl_gset:NV \pnhdpagefirst \l__postnotes_tmpa_tl }
1348   { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
1349 \prop_get:NnNTF \g__postnotes_header_page_last_prop
1350   {#1} \l__postnotes_tmpa_tl
1351   {
1352     \tl_gset:NV \pnhdpagelast \l__postnotes_tmpa_tl
1353     \tl_gset:NV \g__postnotes_header_prev_last_page_tl
1354     \l__postnotes_tmpa_tl
1355   }
1356   { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
1357 \prop_get:NnNTF \g__postnotes_header_chap_first_prop
1358   {#1} \l__postnotes_tmpa_tl
1359   { \tl_gset:NV \pnhdchapfirst \l__postnotes_tmpa_tl }
1360   { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
1361 \prop_get:NnNTF \g__postnotes_header_chap_last_prop
1362   {#1} \l__postnotes_tmpa_tl
1363   {
1364     \tl_gset:NV \pnhdchaplast \l__postnotes_tmpa_tl
1365     \tl_gset:NV \g__postnotes_header_prev_last_chap_tl
1366     \l__postnotes_tmpa_tl
1367   }
1368   { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
1369 \prop_get:NnNTF \g__postnotes_header_sect_first_prop
1370   {#1} \l__postnotes_tmpa_tl
1371   { \tl_gset:NV \pnhdsectfirst \l__postnotes_tmpa_tl }
1372   { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
1373 \prop_get:NnNTF \g__postnotes_header_sect_last_prop
1374   {#1} \l__postnotes_tmpa_tl
1375   {
1376     \tl_gset:NV \pnhdsectlast \l__postnotes_tmpa_tl
1377     \tl_gset:NV \g__postnotes_header_prev_last_sect_tl
1378     \l__postnotes_tmpa_tl
1379   }
1380   { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
1381 \prop_get:NnNTF \g__postnotes_header_name_first_prop
1382   {#1} \l__postnotes_tmpa_tl
1383   { \tl_gset:NV \pnhdnamefirst \l__postnotes_tmpa_tl }
1384   { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
1385 \prop_get:NnNTF \g__postnotes_header_name_last_prop
1386   {#1} \l__postnotes_tmpa_tl
1387   {
1388     \tl_gset:NV \pnhdnamelast \l__postnotes_tmpa_tl
1389     \tl_gset:NV \g__postnotes_header_prev_last_name_tl
1390     \l__postnotes_tmpa_tl
1391   }
1392   { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
1393 \group_end:
1394 }
1395 \cs_generate_variant:Nn \l__postnotes_set_headers_vars:n { e }

```

(End of definition for \l\_\_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.

```

1396 \AddToHook { shipout/before } [ ./header ]
1397   { \__postnotes_set_headers_vars_next: }
1398 \bool_new:N \g__postnotes_header_vars_next_bool
1399 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:
1400   {
1401     \bool_if:NT \g__postnotes_header_vars_next_bool
1402       { \__postnotes_set_headers_vars:e { \int_eval:n { \c@page + 1 } } }
1403   }
1404 \cs_new_protected:Npn \__postnotes_set_headers_vars_first:
1405   {
1406     \__postnotes_set_print_page_label:e
1407       { \int_use:N \g__postnotes_print_postnotes_int }
1408     \__postnotes_set_headers_vars:e
1409       {
1410         \__postnotes_extract_pageref:e
1411           { print@ \int_use:N \g__postnotes_print_postnotes_int }
1412       }
1413   }

```

*(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
1414 \NewDocumentCommand \pnheaderdefault {}
1415   {
1416     \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
1417       { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
1418       { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
1419   }

```

*(End of definition for `\pnheaderdefault`.)*

## 9 Compatibility

A dedicated temp variable for restoring data.

```

1420 \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.

```
1421 \AddToHook { postnotes/note/begin } [ ./compat/caption ]
1422 { \cs_if_exist:NT \@capytype { \postnotesetup { maybemulti } } }
```

## **biblatex**

Thanks Moritz Wemheuer: [https://tex.stackexchange.com/q/597359#comment1594585\\_597389](https://tex.stackexchange.com/q/597359#comment1594585_597389).

```
1423 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1424 {
```

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

```
1425 \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1426 { \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.

```
1427 \AddToHook { begindocument/before } [ postnotes/compat/biblatex ]
1428 {
1429 \cs_set:Npn \blx@theendnote { \postnote }
1430 \cs_set:Npn \blx@theendnotetext { \blx@err@endnote \footnotetext }
1431 }
1432 }
1433 <*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):

```
1434 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1435 {
```

Store biblatex variables for each note.

```

1436 \AddToHook { postnotes/note/store } [ postnotes/compat/biblatex ]
1437 {
1438   \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1439   { biblatex@refsection } { \int_use:N \c@refsection }
1440   \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1441   { biblatex@refsegment } { \int_use:N \c@refsegment }
1442   \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1443   { biblatex@refcontextbool }
1444   { \iftoggle { blx@refcontext } { true } { false } }
1445   \prop_gput:cnV { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1446   { biblatex@refcontext } \blx@refcontext@context
1447 }

```

biblatex setup, once for \printpostnotes call.

```

1448 \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1449 {
1450   \__postnotes_biblatex_endrefcontext_local:
1451   \__postnotes_biblatex_citereset_local:
1452 }

```

Restore biblatex variables for each note.

```

1453 \AddToHook { postnotes/print/note/begin } [ postnotes/compat/biblatex ]
1454 {
1455   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1456   { biblatex@refsection } \l__postnotes_restore_tmp_tl
1457   \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
1458   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1459   { biblatex@refsegment } \l__postnotes_restore_tmp_tl
1460   \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
1461   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1462   { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
1463   \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
1464   \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1465   { biblatex@refcontext } \l__postnotes_restore_tmp_tl
1466   \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
1467 }

```

Auxiliary functions.

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

```

1468 \cs_new_protected:Npn \__postnotes_biblatex_endrefcontext_local:
1469 {
1470   \togglefalse { blx@refcontext }
1471   \tl_clear:N \blx@refcontext@labelprefix
1472   \tl_clear:N \blx@refcontext@labelprefix@real
1473   \tl_set:Ne \blx@refcontext@sortingtemplatename { \blx@sorting }
1474   \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
1475   \tl_set:Nn \blx@refcontext@uniquenametemplatenamename { global }
1476   \tl_set:Nn \blx@refcontext@labelalphanametemplatenamename { global }
1477   \blx@edef@refcontext
1478   {
1479     \blx@refcontext@sortingtemplatename /
1480     \blx@refcontext@sortingnamekeytemplatename /

```

```

1481         /
1482         \blx@refcontext@uniquenamemplatename /
1483         \blx@refcontext@labelalphanamemplatename
1484     }
1485 }

```

(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`.

```

1486     \cs_new_protected:Npn \__postnotes_biblatex_citereset_local:
1487     {
\global\cslet{blx@bsee@the\c@refsection}\@empty
\global\cslet{blx@fsee@the\c@refsection}\@empty
1488         \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1489         \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
\blx@ibidreset@force
1490         \undef \blx@lastkey@text
1491         \undef \blx@lastkey@foot
\blx@idemreset@force
1492         \undef \blx@lasthash@text
1493         \undef \blx@lasthash@foot
\blx@opcitreset@force
1494         \clist_map_inline:Nn \blx@trackhash@text
1495         { \csundef { blx@lastkey@text@ ##1 } }
1496         \tl_clear:N \blx@trackhash@text
1497         \clist_map_inline:Nn \blx@trackhash@foot
1498         { \csundef { blx@lastkey@foot@ ##1 } }
1499         \tl_clear:N \blx@trackhash@foot
\blx@loccitreset@force
1500         \clist_map_inline:Nn \blx@trackkeys@text
1501         { \csundef { blx@lastnote@text@ ##1 } }
1502         \tl_clear:N \blx@trackkeys@text
1503         \clist_map_inline:Nn \blx@trackkeys@foot
1504         { \csundef { blx@lastnote@foot@ ##1 } }
1505         \tl_clear:N \blx@trackkeys@foot
and all of them do:
1506         \cs_set_eq:NN \blx@lastmpfn \z@
1507     }

```

(End of definition for `\__postnotes_biblatex_citereset_local:`)

```

1508 }

```

`biblatex`'s `refsections`, contrary to `refsegments` and `refcontexts` which are handled in the  $\text{\LaTeX}$  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`.

```

1509 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1510   {
1511     \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1512     \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1513     \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1514       {
1515         \tl_set:Ne \l__postnotes_biblatex_orig_refsection_tl
1516           { \int_use:N \c@refsection }
1517         \tl_gset:Ne \g__postnotes_biblatex_prev_refsection_tl
1518           { \l__postnotes_biblatex_orig_refsection_tl }
1519       }
1520     \AddToHook { postnotes/print/note/begin } [ postnotes/compat/biblatex ]
1521       {
1522         \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1523           { biblatex@refsection } \l__postnotes_restore_tmp_tl
1524         \tl_if_eq:N NF
1525           \l__postnotes_restore_tmp_tl
1526           \g__postnotes_biblatex_prev_refsection_tl
1527         {
1528           \int_set:Nn \c@blx@maxsection
1529             { \l__postnotes_restore_tmp_tl - 1 }
1530           \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1531             \l__postnotes_restore_tmp_tl
1532           \group_begin:
1533             \cs_set_eq:NN \label \use_none:n
1534             \cs_set_eq:NN \blx@info \use_none:n
1535             \blx@endrefsection
1536             \refsection
1537           \group_end:
1538         }
1539       }
1540   }
1541 </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.

```

1542 \str_new:N \l__postnotes_note_zlabel_str

```

(End of definition for `\l__postnotes_note_zlabel_str`.)

```

1543 \AddToHook { package/zref-user/after } [ ./compat/zref-user ]
1544 {

```

Provide zlabel option.

```

1545 \keys_define:nn { postnotes/note }
1546 {
1547     zlabel .str_set:N = \l__postnotes_note_zlabel_str ,
1548     zlabel .value_required:n = true ,
1549 }

```

Provide property to store the mark for measuring passes.

```

1550 \zref@newprop { postnote@mark } [] { \l__postnotes_mark_tl }
1551 \AddToHook { postnotes/note/begin } [ postnotes/compat/zref-user ]
1552 { \zref@localaddprop { main } { postnote@mark } }

```

`\postnotezref` Provide `\postnotezref`.

```

\postnotezref*{<label>}

```

```

1553 \NewDocumentCommand \postnotezref { s m }
1554 { \__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>}

```

```

1555 \str_new:N \l__postnotes_note_zref_zlabel_str
1556 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1557 {
1558     \group_begin:
1559     \str_set:Nn \l__postnotes_note_zref_zlabel_str {#2}
1560     \__postnotes_typeset_mark_wrapper:nnn
1561     {
1562         \bool_lazy_all:nTF
1563         {
1564             { ! #1 }
1565             { \l__postnotes_hyperlink_bool }
1566             { \l__postnotes_zrefhyperref_bool }
1567         }
1568         {
1569             \hyperlink
1570             { \zref@extractdefault {#2} { anchor } { } }
1571             { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1572         }
1573         { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1574     }
1575     { \tag_socket_use:n { postnotes/postnotezref/begin } }
1576     { \tag_socket_use:n { postnotes/postnotezref/end } }
1577     \group_end:
1578 }

```

*(End of definition for \\_\_postnotes\_note\_zref:nn.)*

```

1579 }

```

```

1580 \bool_new:N \l__postnotes_zrefhyperref_bool
1581 \AddToHook { package/zref-hyperref/after } [ ./compat/zref-hyperref ]
1582   { \bool_set_true:N \l__postnotes_zrefhyperref_bool }

```

## zref-clever

```

1583 \AddToHook { package/zref-clever/after } [ ./compat/zref-clever ]
1584   {
1585     \zcsetup
1586     {
1587       countertype = { postnote = endnote } ,
1588       countertype = { postnotetext = endnote } ,
1589     }
1590     \AddToHook { postnotes/print/begin } [ postnotes/compat/zref-clever ]
1591     { \zcsetup { counterresetby = { postnotetext = postnotesection } } }
1592   }

```

## zref-check

```

1593 \AddToHook { package/zref-check/after } [ ./compat/zref-check ]
1594   {
1595     \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1596     {
1597       \AddToHook { postnotes/note/store } [ postnotes/compat/zref-check ]
1598       {
1599         \prop_gput:cne { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1600         { zref-check@abschap } { \int_use:N \c@zc@abschap }
1601         \prop_gput:cne { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1602         { zref-check@abssec } { \int_use:N \c@zc@abssec }
1603       }
1604       \AddToHook { postnotes/print/note/begin } [ postnotes/compat/zref-check ]
1605       {
1606         \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1607         { zref-check@abschap } \l__postnotes_restore_tmp_tl
1608         \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1609         \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1610         { zref-check@abssec } \l__postnotes_restore_tmp_tl
1611         \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1612       }
1613     }
1614   }
1615 }

```

## amsmath

```

1616 \AddToHook { package/amsmath/after } [ ./compat/amsmath ]
1617   {

```

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

```

1618     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/amsmath ]
1619     {
1620       \legacy_if:nT { measuring@ }
1621       {
1622         \bool_set_true:N \l__postnotes_inhibit_note_bool
1623         \bool_set_true:N \l__postnotes_print_plain_mark_bool

```



```

1624         \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1625     }
1626 }

```

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.)”

```

1627     \AddToHook { postnotes/note/begin } [ postnotes/compat/amsmath ]
1628     { \legacy_if:nF { firstchoice@ } { \postnotessetup { maybemulti } } }
1629 }

```

## csquotes

```

1630 \AddToHook { package/csquotes/after } [ ./compat/csquotes ]
1631 {
1632     \bool_new:N \l__postnotes_csquotes_measuring_bool
1633     \BlockquoteDisable
1634     { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1635     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/csquotes ]
1636     {
1637         \bool_if:NT \l__postnotes_csquotes_measuring_bool
1638         {
1639             \bool_set_true:N \l__postnotes_inhibit_note_bool
1640             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1641             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1642         }
1643     }
1644 }

```

## 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>.

```

1645 \AddToHook { package/tabularx/after } [ ./compat/tabularx ]
1646 {
1647     \bool_new:N \l__postnotes_tabularx_inside_env_bool
1648     \AddToHook { env/tabularx/begin } [ postnotes/compat/tabularx ]
1649     {
1650         \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1651         \cs_set_eq:NN \__postnotes_tabularx_saved_write:nN \write
1652     }
1653     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/tabularx ]
1654     {
1655         \bool_lazy_and:nnT

```

```

1656     { \l__postnotes_tabularx_inside_env_bool }
1657     { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1658     {
1659         \bool_set_true:N \l__postnotes_inhibit_note_bool
1660         \bool_set_true:N \l__postnotes_print_plain_mark_bool
1661         \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1662     }
1663 }
1664 \AddToHook { package/xltabular/after } [ postnotes/compat/xltabular ]
1665 {
1666     \AddToHook { env/xltabular/begin } [ postnotes/compat/xltabular ]
1667     {
1668         \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1669         \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1670     }
1671 }
1672 }

```

## tabularray

```

1673 \AddToHook { package/tabularray/after } [ ./compat/tabularray ]
1674 {

```

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).

```

1675     \bool_if_exist:NT \lTblrMeasuringBool
1676     {

```

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.

```

1677     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/tabularray ]
1678     {
1679         \bool_if:NT \lTblrMeasuringBool
1680         {
1681             \bool_set_true:N \l__postnotes_inhibit_note_bool
1682             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1683             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1684         }
1685     }
1686 }
1687 }

```

## 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.

```
1688 \socket_new:nm { tagsupport/postnotes/mark/begin }{ 0 }
```

```

1689 \socket_new:nn { tagsupport/postnotes/mark/end }{ 0 }
1690 \socket_new:nn { tagsupport/postnotes/nomark/begin }{ 0 }
1691 \socket_new:nn { tagsupport/postnotes/nomark/end }{ 0 }
1692 \socket_new:nn { tagsupport/postnotes/multisep/begin }{ 0 }
1693 \socket_new:nn { tagsupport/postnotes/multisep/end }{ 0 }
1694 \socket_new:nn { tagsupport/postnotes/printlist/begin }{ 0 }
1695 \socket_new:nn { tagsupport/postnotes/printlist/end }{ 0 }
1696 \socket_new:nn { tagsupport/postnotes/printnote/begin }{ 0 }
1697 \socket_new:nn { tagsupport/postnotes/printnote/end }{ 0 }
1698 \socket_new:nn { tagsupport/postnotes/printmark/begin }{ 0 }
1699 \socket_new:nn { tagsupport/postnotes/printmark/end }{ 0 }
1700 \socket_new:nn { tagsupport/postnotes/printtext/begin }{ 0 }
1701 \socket_new:nn { tagsupport/postnotes/printtext/end }{ 0 }
1702 \socket_new:nn { tagsupport/postnotes/postnoteref/begin }{ 0 }
1703 \socket_new:nn { tagsupport/postnotes/postnoteref/end }{ 0 }
1704 \socket_new:nn { tagsupport/postnotes/postnoterezref/begin }{ 0 }
1705 \socket_new:nn { tagsupport/postnotes/postnoterezref/end }{ 0 }

1706 \bool_lazy_and:nnT
1707   { \cs_if_exist_p:N \tag_if_active_p: }
1708   { \tag_if_active_p: }
1709   {

```

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

```

1710   \tagpdfsetup
1711   {
1712     role/new-tag = { tag=endnote, role=FENote } ,
1713     role/new-tag = { tag=endnotemark, role=Lbl } ,
1714     role/new-tag = { tag=endnotelabel, role=Lbl } ,
1715     role/new-attribute =
1716       { EndnoteType } { /0 /FENote /NoteType /Endnote } ,
1717     role/new-attribute =
1718       { EndnotesList } { /0 /List /ListNumbering /Ordered } ,
1719   }

\postnote

1720   \socket_new_plug:nnn { tagsupport/postnotes/mark/begin } { default }
1721   {
1722     \tag_mc_end_push:
1723     \tag_struct_begin:n
1724     {
1725       tag = endnotemark ,
1726       label = { postnotemark. \l_postnotes_note_id_tl } ,
1727       ref = { postnote. \l_postnotes_note_id_tl } ,
1728     }
1729     \__postnotes_tagsup_store_sstructnum:nN
1730     { postnotemark } \l_postnotes_note_id_tl
1731     \tag_mc_begin:n { }
1732   }
1733   \socket_new_plug:nnn { tagsupport/postnotes/mark/end } { default }
1734   {
1735     \tag_mc_end:
1736     \tag_struct_end: % endnotemark
1737     \tag_mc_begin_pop:n { }

```

```

1738     }
1739     \socket_new_plug:nmn { tagssupport/postnotes/nomark/begin } { default }
1740     {
1741         \tag_struct_begin:n
1742         {
1743             tag = NonStruct ,
1744             label = { postnotemark. \l_postnotes_note_id_tl } ,
1745             ref = { postnote. \l_postnotes_note_id_tl } ,
1746         }
1747         \l_postnotes_tagsup_store_sctructnum:nN
1748         { postnotemark } \l_postnotes_note_id_tl
1749     }
1750     \socket_new_plug:nmn { tagssupport/postnotes/nomark/end } { default }
1751     { \tag_struct_end: } % NonStruct
1752     \socket_assign_plug:nm { tagssupport/postnotes/mark/begin } { default }
1753     \socket_assign_plug:nm { tagssupport/postnotes/mark/end } { default }
1754     \socket_assign_plug:nm { tagssupport/postnotes/nomark/begin } { default }
1755     \socket_assign_plug:nm { tagssupport/postnotes/nomark/end } { default }
multiple
1756     \socket_new_plug:nmn { tagssupport/postnotes/multisep/begin } { default }
1757     {
1758         \tag_mc_end_push:
1759         \tag_mc_begin:n { artifact }
1760     }
1761     \socket_new_plug:nmn { tagssupport/postnotes/multisep/end } { default }
1762     {
1763         \tag_mc_end:
1764         \tag_mc_begin_pop:n { }
1765     }
1766     \socket_assign_plug:nm { tagssupport/postnotes/multisep/begin } { default }
1767     \socket_assign_plug:nm { tagssupport/postnotes/multisep/end } { default }
\printpostnotes
1768     \socket_new_plug:nmn { tagssupport/postnotes/printlist/begin } { default }
1769     { \tag_tool:n { para/tagging=false } }
1770     \socket_new_plug:nmn { tagssupport/postnotes/printlist/end } { default }
1771     { }
1772     \socket_assign_plug:nm { tagssupport/postnotes/printlist/begin } { default }
1773     \socket_assign_plug:nm { tagssupport/postnotes/printlist/end } { default }
1774     \socket_new_plug:nmn { tagssupport/postnotes/printnote/begin } { default }
1775     {
1776         \bool_if:NF \l__postnotes_print_as_list_bool
1777         {
1778             \tag_struct_begin:n
1779             {
1780                 tag = endnote ,
1781                 attribute-class = EndnoteType ,
1782                 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.
1783                 ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1784             }
1785             \l_postnotes_tagsup_store_sctructnum:nN

```

```

1786         { postnote } \l_postnotes_print_note_id_tl
1787     }
1788 }
1789 \socket_new_plug:nnn { tagssupport/postnotes/printnote/end } { default }
1790 {
1791     \bool_if:NF \l__postnotes_print_as_list_bool
1792     { \tag_struct_end: } % endnote
1793 }
1794 \socket_assign_plug:nn { tagssupport/postnotes/printnote/begin } { default }
1795 \socket_assign_plug:nn { tagssupport/postnotes/printnote/end } { default }
1796 \socket_new_plug:nnn { tagssupport/postnotes/printmark/begin } { default }
1797 {
1798     \bool_if:NF \l__postnotes_print_as_list_bool
1799     {
1800         \tag_struct_begin:n { tag=endnotelabel }
1801         \tag_mc_begin:n { tag=Lbl }
1802     }
1803 }
1804 \socket_new_plug:nnn { tagssupport/postnotes/printmark/end } { default }
1805 {
1806     \bool_if:NF \l__postnotes_print_as_list_bool
1807     {
1808         \tag_mc_end:
1809         \tag_struct_end: % endnotelabel
1810     }
1811 }
1812 \socket_assign_plug:nn { tagssupport/postnotes/printmark/begin } { default }
1813 \socket_assign_plug:nn { tagssupport/postnotes/printmark/end } { default }
1814 \socket_new_plug:nnn { tagssupport/postnotes/printtext/begin } { default }
1815 {
1816     \bool_if:NTF \l__postnotes_print_as_list_bool
1817     {
1818         \tag_struct_begin:n
1819         {
1820             tag = endnote ,
1821             attribute-class = EndnoteType ,
1822             label = { postnote. \l_postnotes_print_note_id_tl } ,

```

CHECK Ditto.

```

1823         ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1824     }
1825     \__postnotes_tagsup_store_sctructnum:nN
1826     { postnote } \l_postnotes_print_note_id_tl
1827     \tag_struct_begin:n { tag=text-unit }
1828     \tag_struct_begin:n { tag=text }
1829     \tag_tool:n { para/tagging=true }
1830     \tag_mc_begin:n { }
1831 }
1832 {
1833     \tag_struct_begin:n { tag=text-unit }
1834     \tag_struct_begin:n { tag=text }
1835     \tag_tool:n { para/tagging=true }
1836     \tag_mc_begin:n { }
1837 }
1838 }

```

```

1839 \socket_new_plug:nmn { tagsupport/postnotes/printtext/end } { default }
1840 {
1841   \bool_if:NTF \l__postnotes_print_as_list_bool
1842   {
1843     \tag_mc_end:
1844     \tag_tool:n { para/tagging=false }
1845     \tag_struct_end: % text
1846     \tag_struct_end: % text-unit
1847     \tag_struct_end: % endnote
1848   }
1849   {
1850     \tag_mc_end:
1851     \tag_tool:n { para/tagging=false }
1852     \tag_struct_end: % text
1853     \tag_struct_end: % text-unit
1854   }
1855 }
1856 \socket_assign_plug:nm { tagsupport/postnotes/printtext/begin } { default }
1857 \socket_assign_plug:nm { 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.

```

1858 \IfInstanceExistsT { blockenv } { list }
1859 {
1860   \DeclareInstance { blockenv } { postnoteslist } { display }
1861   {
1862     env-name      = postnoteslist ,
1863     tag-name      = L ,
1864     tag-class     = EndnotesList ,
1865     tagging-recipe = list ,
1866     inner-level-counter = ,
1867     level-increase = true ,
1868     setup-code    = ,
1869     block-instance = list ,
1870     inner-instance = postnoteslist ,
1871   }
1872   \DeclareInstanceCopy { blockenv }
1873   { postnoteslisthang } { postnoteslist }
1874   \EditInstance { blockenv } { postnoteslisthang }
1875   { env-name = postnoteslisthang }
1876   \DeclareInstance { list } { postnoteslist } { std }
1877   { item-instance = postnoteslist }
1878   \DeclareInstance { item } { postnoteslist } { std }
1879   {
1880     label-format = { \hspace { \labelsep } \normalfont ~ #1 } ,
1881     label-align = left ,
1882   }
1883   \RenewDocumentEnvironment { postnoteslist } { }
1884   {
1885     \UseInstance { blockenv } { postnoteslist }
1886     {
1887       leftmargin    = Opt ,
1888       label-width   = Opt ,

```



```

1889         item-indent    = .5\parindent ,
1890         rightmargin    = Opt ,
1891         parindent      = \parindent ,
1892         par-skip       = \parskip ,
1893         item-skip      = Opt ,
1894         beginsep       = .5\topsep ,
1895         begin-par-skip = .5\partopsep ,
1896     }
1897 }
1898 { \endblockenv }
1899 \RenewDocumentEnvironment { postnoteslisthang } { }
1900 {
1901     \UseInstance { blockenv } { postnoteslisthang }
1902     {
1903         leftmargin    = 1em ,
1904         label-width   = -\leftmargin ,
1905         item-indent   = -2\leftmargin ,
1906         rightmargin   = Opt ,
1907         parindent     = \parindent ,
1908         par-skip      = \parskip ,
1909         item-skip     = Opt ,
1910         beginsep      = .5\topsep ,
1911         begin-par-skip = .5\partopsep ,
1912     }
1913 }
1914 { \endblockenv }
1915 }

```

Setup for \label and \zlabel inside the note.

```

1916 \bool_new:N \l__postnotes_inside_note_bool
1917 \AddToHookWithArguments { label } [ postnotes/tagsup ]
1918 {
1919     \bool_if:NT \l__postnotes_inside_note_bool
1920     {
1921         \property_record:nn { postnote@label@innote. #1 }
1922         { postnotes/tagsup@noteid }
1923     }
1924 }
1925 \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1926 { \bool_set_true:N \l__postnotes_inside_note_bool }
1927 \property_new:nnnn { postnotes/tagsup@noteid } { now } { 0 }
1928 { \l__postnotes_print_note_id_tl }
1929 \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
1930 {
1931     \zref@newprop { postnotes@tagsup@noteid } [ 0 ]
1932     { \l__postnotes_print_note_id_tl }
1933     \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1934     { \zref@localaddprop { main } { postnotes@tagsup@noteid } }
1935 }

```

Setup for label and zlabel options.

```

1936 \AddToHook { postnotes/note/store } [ postnotes/tagsup ]
1937 {
1938     \str_if_empty:NF \l__postnotes_note_label_str
1939     {

```

```

1940         \prop_gput:cnV
1941         { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1942         { label } \l__postnotes_note_label_str
1943     }
1944 }
1945 \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
1946 {
1947     \AddToHook { postnotes/note/store } [ postnotes/tagsup ]
1948     {
1949         \str_if_empty:NF \l__postnotes_note_zlabel_str
1950         {
1951             \prop_gput:cnV
1952             { \_postnotes_data_name:e { \l_postnotes_note_id_tl } }
1953             { zlabel } \l__postnotes_note_zlabel_str
1954         }
1955     }
1956 }
1957 \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1958 {
1959     \_postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1960     { label } \l__postnotes_restore_tmp_tl
1961     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1962     {
1963         \exp_args:Ne \property_record:nn
1964         { postnote@label@option. \l__postnotes_restore_tmp_tl }
1965         { postnotes/tagsup@noteid }
1966     }
1967     \_postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1968     { zlabel } \l__postnotes_restore_tmp_tl
1969     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1970     {
1971         \exp_args:Ne \property_record:nn
1972         { postnote@zlabel@option. \l__postnotes_restore_tmp_tl }
1973         { postnotes/tagsup@noteid }
1974     }
1975 }

```

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_sctructnum: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".

```

```

1976 \prop_new:N \g__postnotes_tagsup_structnums_prop
1977 \cs_new_protected:Npn \__postnotes_tagsup_store_sctructnum:nN #1#2
1978 {
1979     \prop_gput:Nee \g__postnotes_tagsup_structnums_prop
1980     { #1 . #2 } { \tag_get:n { struct_num } }
1981 }
1982 \prop_new:N \g__postnotes_tagsup_crossrefs_prop
1983 \cs_new_protected:Npn \__postnotes_tagsup_store_crossref:nN #1#2

```

```

1984     {
1985     \prop_gput:Nee \g__postnotes_tagsup_crossrefs_prop
1986     { \tag_get:n { struct_num } } { #1 . #2 }
1987     }

```

(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.

```

```

1988     \cs_new_protected:Npn \__postnotes_tagsup_gput_ref:nn #1#2
1989     {
1990     \tag_if_active:T
1991     { \tag_struct_gput:ene {#1} {ref} { \tag_struct_object_ref:e {#2} } }
1992     }

```

(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.

```

1993     \AddToHook { tagpdf/finish/before } [ postnotes/tagsup ]
1994     {
1995     \prop_map_inline:Nn \g__postnotes_tagsup_crossrefs_prop
1996     {
1997     \__postnotes_tagsup_gput_ref:nn
1998     { \prop_item:Nn \g__postnotes_tagsup_structnums_prop {#2} }
1999     {#1}
2000     }
2001     }

```

`\postnoteref`

```

2002     \socket_new_plug:nnn { tagssupport/postnotes/postnoteref/begin } { default }
2003     {
2004     \tag_mc_end_push:
2005     \property_if_recorded:eeTF
2006     { postnote@label@innote. \l__postnotes_note_ref_label_str }
2007     { postnotes/tagsup@noteid }
2008     {

```

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

```

2009     \tl_set:Nc \l__postnotes_tmpa_tl
2010     {
2011     \property_ref:ee
2012     { postnote@label@innote. \l__postnotes_note_ref_label_str }
2013     { postnotes/tagsup@noteid }
2014     }
2015     \tag_struct_begin:n
2016     {
2017     tag = endnotemark ,
2018     ref = { postnote. \l__postnotes_tmpa_tl } ,
2019     }
2020     \__postnotes_tagsup_store_crossref:nN

```

```

2021         { postnote } \l__postnotes_tmpa_tl
2022     }
2023     {
2024         \property_if_recorded:eeTF
2025         { postnote@label@option. \l__postnotes_note_ref_label_str }
2026         { postnotes/tagsup@noteid }
2027     }

```

Label coming from a label option.

```

2028         \tl_set:Ne \l__postnotes_tmpa_tl
2029         {
2030             \property_ref:ee
2031             { postnote@label@option. \l__postnotes_note_ref_label_str }
2032             { postnotes/tagsup@noteid }
2033         }
2034         \tag_struct_begin:n
2035         {
2036             tag = endnotemark ,
2037             ref = { postnotemark. \l__postnotes_tmpa_tl } ,
2038         }
2039         \__postnotes_tagsup_store_crossref:nN
2040         { postnotemark } \l__postnotes_tmpa_tl
2041     }
2042     { \tag_struct_begin:n { tag = endnotemark } }
2043 }
2044 \tag_mc_begin:n { }
2045 }
2046 \socket_new_plug:nnn { tagssupport/postnotes/postnoteref/end } { default }
2047 {
2048     \tag_mc_end:
2049     \tag_struct_end: % endnotemark
2050     \tag_mc_begin_pop:n { }
2051 }
2052 \socket_assign_plug:nn { tagssupport/postnotes/postnoteref/begin } { default }
2053 \socket_assign_plug:nn { tagssupport/postnotes/postnoteref/end } { default }

```

\postnotezref

```

2054 \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
2055 {
2056     \socket_new_plug:nnn { tagssupport/postnotes/postnotezref/begin } { default }
2057     {
2058         \tag_mc_end_push:
2059         \zref@ifrefcontainsprop { \l__postnotes_note_zref_zlabel_str }
2060         { postnotes@tagsup@noteid }
2061     }

```

Label coming from a \zlabel inside the note.

```

2062         \tl_set:Ne \l__postnotes_tmpa_tl
2063         {
2064             \zref@extract { \l__postnotes_note_zref_zlabel_str }
2065             { postnotes@tagsup@noteid }
2066         }
2067         \tag_struct_begin:n
2068         {
2069             tag = endnotemark ,

```

```

2070         ref = { postnote. \l__postnotes_tmpa_tl } ,
2071     }
2072     \__postnotes_tagsup_store_crossref:nN
2073     { postnote } \l__postnotes_tmpa_tl
2074 }
2075 {
2076     \property_if_recorded:eeTF
2077     { postnote@zlabel@option. \l__postnotes_note_zref_zlabel_str }
2078     { postnotes/tagsup@noteid }
2079     {

```

Label coming from a zlabel option.

```

2080     \tl_set:Nc \l__postnotes_tmpa_tl
2081     {
2082         \property_ref:ee
2083         {
2084             postnote@zlabel@option.
2085             \l__postnotes_note_zref_zlabel_str
2086         }
2087         { postnotes/tagsup@noteid }
2088     }
2089     \tag_struct_begin:n
2090     {
2091         tag = endnotemark ,
2092         ref = { postnotemark. \l__postnotes_tmpa_tl } ,
2093     }
2094     \__postnotes_tagsup_store_crossref:nN
2095     { postnotemark } \l__postnotes_tmpa_tl
2096 }
2097 { \tag_struct_begin:n { tag = endnotemark } }
2098 }
2099 \tag_mc_begin:n { }
2100 }
2101 \socket_new_plug:nnn { tagsupport/postnotes/postnotezref/end } { default }
2102 {
2103     \tag_mc_end:
2104     \tag_struct_end: % endnotemark
2105     \tag_mc_begin_pop:n { }
2106 }
2107 \socket_assign_plug:nn { tagsupport/postnotes/postnotezref/begin } { default }
2108 \socket_assign_plug:nn { tagsupport/postnotes/postnotezref/end } { default }
2109 }
2110 }

```

## 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 2111 \tl_new:N \pntitle
\pnhdtopage 2112 \tl_new:N \pnhdnotes
\pnhdtopages 2113 \tl_new:N \pnhdtopage
2114 \tl_new:N \pnhdtopages
2115 \tl_set:Nn \pntitle { Notes }

```

```

2116 \tl_set:Nn \pnhdnotes { Notes }
2117 \tl_set:Nn \pnhdtopage { to~page }
2118 \tl_set:Nn \pnhdtopages { to~pages }

```

(End of definition for \pntitle and others.)

`\__postnotes_define_language:nn` Defines language specific values for `\langle postnote language \rangle` by storing a set of assignments for the language specific variables in `\langle setup \rangle`. `\langle postnote language \rangle` 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:nn {\langle postnote language \rangle} {\langle setup \rangle}

2119 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
2120 {
2121   \tl_new:c { g__postnotes_language_ #1 _tl }
2122   \tl_gset:cn { g__postnotes_language_ #1 _tl } {#2}
2123 }

```

(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/\langle language \rangle/beforeextras` so that users can override it with a traditional `\addto\extras\langle language \rangle`.

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 – `\langle language \rangle\langle caption \rangle`, which is meant to be set with `\setlocalecaption`, and not suitable for our needs. The `\extras\langle language \rangle` 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 `\langle babel language \rangle` to execute the setup defined by `\__postnotes_define_language:nn` for `\langle postnote language \rangle` at the `babel/\langle language \rangle/beforeextras` hook.

```

\__postnotes_set_babel_language:nn {\langle babel language \rangle} {\langle postnote language \rangle}

2124 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
2125 {
2126   \ActivateGenericHook { babel/#1/beforeextras }
2127   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
2128   { g__postnotes_language_ #2 _tl }
2129 }

```

(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)}

2130 \cs_new_protected:Npn \_postnotes_set_polyglossia_language:nn #1#2
2131 {
2132   \AddToHook { package/polyglossia/after }
2133   {
2134     \exp_args:Nnv \csgappto { captions #1 }
2135     { g__postnotes_language_ #2 _tl }
2136   }
2137 }
```

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

## English

```
2138 \_postnotes_define_language:nn { english }
2139 {
2140   \tl_set:Nn \pntitle      { Notes }
2141   \tl_set:Nn \pnhdnotes   { Notes }
2142   \tl_set:Nn \pnhdtopage  { to~page }
2143   \tl_set:Nn \pnhdtopages { to~pages }
2144 }
2145 \_postnotes_set_babel_language:nn { english } { english }
2146 \_postnotes_set_babel_language:nn { british } { english }
2147 \_postnotes_set_babel_language:nn { american } { english }
2148 \_postnotes_set_babel_language:nn { canadian } { english }
2149 \_postnotes_set_babel_language:nn { australian } { english }
2150 \_postnotes_set_babel_language:nn { newzealand } { english }
2151 \_postnotes_set_babel_language:nn { UKenglish } { english }
2152 \_postnotes_set_babel_language:nn { USenglish } { english }
2153 \_postnotes_set_polyglossia_language:nn { english } { english }
```

## Portuguese

```
2154 \_postnotes_define_language:nn { portuguese }
2155 {
2156   \tl_set:Nn \pntitle      { Notas }
2157   \tl_set:Nn \pnhdnotes   { Notas }
2158   \tl_set:Nn \pnhdtopage  { da~página }
```

```

2159     \tl_set:Nn \pnhdtopages { das-páginas }
2160   }
2161   \__postnotes_set_babel_language:nn { portuguese } { portuguese }
2162   \__postnotes_set_babel_language:nn { brazilian } { portuguese }
2163   \__postnotes_set_babel_language:nn { portuges } { portuguese }
2164   \__postnotes_set_babel_language:nn { brazil } { portuguese }
2165   \__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`.

```

2166 \__postnotes_define_language:nn { french }
2167   {
2168     \tl_set:Nn \pntitle { Notes }
2169     \tl_set:Nn \pnhdnotes { Notes }
2170     \tl_set:Nn \pnhdtopage { de-la-page }
2171     \tl_set:Nn \pnhdtopages { des-pages }
2172   }
2173 \__postnotes_set_babel_language:nn { french } { french }
2174 \__postnotes_set_babel_language:nn { acadian } { french }
2175 \__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`.

```

2176 \__postnotes_define_language:nn { german }
2177   {
2178     \tl_set:Nn \pntitle { Endnoten }
2179     \tl_set:Nn \pnhdnotes { Endnoten }
2180     \tl_set:Nn \pnhdtopage { zu-Seite }
2181     \tl_set:Nn \pnhdtopages { zu-Seiten }
2182   }
2183 \__postnotes_set_babel_language:nn { german } { german }
2184 \__postnotes_set_babel_language:nn { ngerman } { german }
2185 \__postnotes_set_babel_language:nn { austrian } { german }
2186 \__postnotes_set_babel_language:nn { naustrian } { german }
2187 \__postnotes_set_babel_language:nn { swissgerman } { german }
2188 \__postnotes_set_babel_language:nn { nswissgerman } { german }
2189 \__postnotes_set_polyglossia_language:nn { german } { german }
2190 </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.

<b>A</b>	<code>\addto</code> . . . . . <a href="#">62</a>
<code>\ActivateGenericHook</code> . . . . .	<a href="#">2126</a>





<b>F</b>		iow commands:	
<code>\fmtversion</code> .....	5	<code>\iow_char:N</code> .....	1000, 1196
fnote internal commands:		<code>\iow_now:Nn</code> .....	122, 127, 132, 143, 148, 153
<code>\_fnote_gput_ref:nn</code> .....	59	<code>\iow_shipout_e:Nn</code> .....	6
<code>\footnote</code> .....	12, 49	<code>\item</code> .....	29, 30, 52, 957
<code>\footnotemark</code> .....	12, 15–17	<code>\itemindent</code> .....	252, 269
<code>\footnotesize</code> .....	313	<code>\itemsep</code> .....	257, 274
<code>\footnotetext</code> .....	15, 16, 1430	<b>K</b>	
<code>\footref</code> .....	58	<code>\kern</code> .....	406, 407
fp commands:		keys commands:	
<code>\fp_compare:nNnTF</code> .....	1163	<code>\keys_define:nn</code> .....	199, 220,
<code>\fp_new:N</code> .....	579	228, 282, 296, 305, 338, 371, 393,	
<code>\fp_set:Nn</code> .....	588	431, 439, 449, 456, 466, 582, 785, 1545	
<code>\fp_use:N</code> .....	36	<code>\keys_set:nn</code> .....	500, 515, 772
<b>G</b>		<b>L</b>	
<code>\gappto</code> .....	62	<code>\label</code> .....	17, 52, 57, 59, 728, 1533
group commands:		<code>\labelsep</code> .....	281, 1880
<code>\group_begin:</code> .....		<code>\labelwidth</code> .....	251, 268
.....	514, 630, 640, 741, 767, 820,	<code>\lastkern</code> .....	13, 416
	878, 919, 923, 1017, 1090, 1127,	<code>\leftmargin</code> .....	250, 267, 268, 269, 1904, 1905
	1146, 1177, 1227, 1344, 1532, 1558	<code>\leftskip</code> .....	315
<code>\group_end:</code> .....		legacy commands:	
.....	575, 634, 646, 756, 781, 897,	<code>\legacy_if:nTF</code> .....	
	974, 985, 997, 1082, 1116, 1137,	..	120, 141, 162, 482, 492, 1620, 1628
	1172, 1192, 1340, 1393, 1537, 1577	<code>\list</code> .....	248, 265
<b>H</b>		<code>\listparindent</code> .....	255, 272
<code>\hbox</code> .....	287	<code>\lTblrMeasuringBool</code> .....	50, 1675, 1679
hbox commands:		<b>M</b>	
<code>\hbox_set:Nn</code> .....	320	<code>\makeatletter</code> .....	63
<code>\hspace</code> .....	281, 1880	<code>\makeatother</code> .....	63
<code>\hyperlink</code> .....	1569	<code>\makelabel</code> .....	253, 270
<code>\hyperref</code> .....	749	<code>\MakeLinkTarget</code> .....	2, 557, 944
<b>I</b>		<code>\mathchoice</code> .....	16, 49
<code>\IfFormatAtLeastTF</code> .....	5, 6	<code>\mbox</code> .....	15
<code>\IfInstanceExistsT</code> .....	1858	<code>\mkbibendnote</code> .....	43
<code>\IfPackageAtLeastTF</code> .....	1595	mode commands:	
<code>\IfPackageLoadedTF</code> .....	364	<code>\mode_if_horizontal:TF</code> .....	709, 720
<code>\iftoggle</code> .....	1444	<code>\mode_leave_vertical:</code> ..	708, 962, 965
int commands:		msg commands:	
<code>\int_compare:nNnTF</code> .....	1037, 1067	<code>\msg_line_context:</code> .....	
<code>\int_eval:n</code> .....	1402	.....	386, 1000, 1120, 1141, 1197
<code>\int_gadd:Nn</code> .....	106, 478	<code>\msg_new:nnn</code> .....	
<code>\int_gincr:N</code> .....	519, 768, 769, 821	.....	385, 387, 999, 1119, 1140, 1194
<code>\int_gset:Nn</code> .....	476	<code>\msg_warning:nn</code> .....	368, 839, 1136
<code>\int_incr:N</code> .....	631	<code>\msg_warning:nnn</code> .....	
<code>\int_new:N</code> .....	86, 502, 506, 764, 801	.....	375, 380, 470, 1109, 1189
<code>\int_set:Nn</code> .....	523, 528,	<code>\multfootsep</code> .....	12
	533, 937, 1457, 1460, 1528, 1608, 1611	<code>\multiplefootnotemarker</code> .....	12
<code>\int_use:N</code> .....	32,	<b>N</b>	
	37, 51, 108, 169, 419, 504, 559,	<code>\NeedsTeXFormat</code> .....	4
	712, 837, 1023, 1407, 1411, 1439,	<code>\newcounter</code> .....	501, 815, 816
	1441, 1488, 1489, 1516, 1600, 1602		



\g\_\_postnotes\_firststrun\_bool . . . . . 88, 89, 113, 548, 825, 1087, 1125  
\\_postnotes\_get\_headers\_data:N . . . . . 36, 38, 40, 848, 1225, 1225  
\\_postnotes\_get\_label\_if\_-exist:N . . . . . 21, 553, 637, 652, 652  
\\_postnotes\_get\_pageref:Nn . . . . . 7, 185, 185, 191, 925, 1260, 1268, 1311  
\g\_\_postnotes\_header\_chap\_first\_-prop . . . . . 1207, 1238, 1298, 1357  
\g\_\_postnotes\_header\_chap\_last\_-prop . . . . . 1207, 1239, 1283, 1330, 1361  
\g\_\_postnotes\_header\_name\_first\_-prop . . . . . 1207, 1242, 1308, 1381  
\g\_\_postnotes\_header\_name\_last\_-prop . . . . . 1207, 1243, 1289, 1336, 1385  
\g\_\_postnotes\_header\_page\_first\_-prop . . . . . 1207, 1236, 1293, 1345  
\g\_\_postnotes\_header\_page\_last\_-prop . . . . . 1207, 1237, 1280, 1327, 1349  
\g\_\_postnotes\_header\_prev\_last\_-chap\_tl 1207, 1245, 1360, 1365, 1368  
\g\_\_postnotes\_header\_prev\_last\_-name\_tl 1207, 1247, 1384, 1389, 1392  
\g\_\_postnotes\_header\_prev\_last\_-page\_tl 1207, 1244, 1348, 1353, 1356  
\g\_\_postnotes\_header\_prev\_last\_-sect\_tl 1207, 1246, 1372, 1377, 1380  
\g\_\_postnotes\_header\_sect\_first\_-prop . . . . . 1207, 1240, 1303, 1369  
\g\_\_postnotes\_header\_sect\_last\_-prop . . . . . 1207, 1241, 1286, 1333, 1373  
\g\_\_postnotes\_header\_vars\_next\_-bool . . . . . 42, 847, 993, 1398, 1401  
\l\_\_postnotes\_hyperlink\_bool 335, 343, 348, 353, 369, 693, 747, 1004, 1565  
\l\_\_postnotes\_hyperref\_warn\_bool . . . . . 336, 344, 349, 354, 367  
\\_postnotes\_inhibit\_note: . . . . . 609  
\\_postnotes\_inhibit\_note:TF . . . . . 19, 33, 517, 605  
\l\_\_postnotes\_inhibit\_note\_bool . . . . . 20, 605, 611, 617, 648, 1622, 1639, 1659, 1681  
\l\_\_postnotes\_inside\_note\_bool . . . . . 1916, 1919, 1926  
\g\_\_postnotes\_labelseq\_seq . . . . . 31, 85, 98, 1024, 1025, 1029, 1054, 1055, 1058, 1080, 1180  
\\_postnotes\_list\_makelabel:n . . . . . 253, 270, 280  
\\_postnotes\_make\_mark:nnn . . . . . 13, 284, 292, 423, 644, 695, 699, 750, 752, 1571, 1573  
\\_postnotes\_make\_text\_mark:nnn . . . . . 288, 1007, 1011  
\l\_\_postnotes\_manual\_sortnum\_-bool . . . . . 35, 581, 589  
\l\_\_postnotes\_mark\_tl . . . . . 21, 30, 520, 531, 538, 542, 577, 584, 622, 639, 735, 1550  
\l\_\_postnotes\_mark\_typeset\_tl . . . . . 502, 542, 553, 570, 627, 635, 637, 639, 644  
\l\_\_postnotes\_maybe\_multi\_bool . . . . . 33, 53, 448, 451, 545, 602  
\l\_\_postnotes\_multiple\_bool . . . . . 391, 395, 404, 413  
\\_postnotes\_multiple\_check: . . . . . 13, 411, 713  
\\_postnotes\_multiple\_prepare: . . . . . 12, 402, 719  
\l\_\_postnotes\_multisep\_tl . . . . . 389, 398, 423  
\l\_\_postnotes\_nomark\_bool . . . . . 516, 562, 574, 578, 598, 619  
\\_postnotes\_note:nn . . . . . 18, 22, 23, 510, 511, 512  
\g\_\_postnotes\_note\_id\_int . . . . . 17, 31, 502, 519, 769, 1040, 1070  
\l\_\_postnotes\_note\_label\_str . . . . . 580, 600, 654, 677, 683, 726, 728, 729, 1938, 1942  
\\_postnotes\_note\_ref:nn . . . . . 24, 737, 738, 739  
\l\_\_postnotes\_note\_ref\_label\_str . . . . . 738, 742, 2006, 2012, 2025, 2031  
\l\_\_postnotes\_note\_set\_labels\_tl . . . . . 502, 555, 565, 571  
\l\_\_postnotes\_note\_zlabel\_str . . . . . 46, 656, 658, 662, 668, 732, 733, 1542, 1547, 1949, 1953  
\\_postnotes\_note\_zref:nn . . . . . 47, 1554, 1555, 1556  
\l\_\_postnotes\_note\_zref\_zlabel\_-str 1555, 1559, 2059, 2064, 2077, 2085  
\c\_\_postnotes\_page\_counter\_tl . . . . . 160  
\l\_\_postnotes\_post\_printnote\_tl . . . . . 235, 295, 302, 973  
\l\_\_postnotes\_post\_textmark\_tl . . . . . 294, 300, 952  
\g\_\_postnotes\_postnote\_counteraux\_-int . . . . . 86, 106, 108, 476, 478  
\l\_\_postnotes\_pre\_textmark\_tl . . . . . 293, 298, 948  
\l\_\_postnotes\_prev\_mark\_chap\_tl . . . . . 1207, 1251, 1271, 1285, 1314, 1332

\l__postnotes_prev_mark_name_tl .	.. <a href="#">1207</a> , <a href="#">1253</a> , <a href="#">1275</a> , <a href="#">1291</a> , <a href="#">1318</a> , <a href="#">1338</a>	887, 890, 893, 911, 927, 930, 933, 1149, 1150, 1153, 1154, 1159, 1161,
\l__postnotes_prev_mark_page_tl .	.. <a href="#">1207</a> , <a href="#">1250</a> , <a href="#">1269</a> , <a href="#">1282</a> , <a href="#">1312</a> , <a href="#">1329</a>	1270, 1272, 1274, 1296, 1301, 1306, 1313, 1315, 1317, 1455, 1458, 1461,
\l__postnotes_prev_mark_sect_tl .	.. <a href="#">1207</a> , <a href="#">1252</a> , <a href="#">1273</a> , <a href="#">1288</a> , <a href="#">1316</a> , <a href="#">1335</a>	1464, 1522, 1606, 1609, 1959, 1967
\l__postnotes_prev_text_page_tl .	..... <a href="#">39</a> , <a href="#">40</a> , <a href="#">1207</a> , <a href="#">1248</a> , <a href="#">1265</a> , <a href="#">1278</a> , <a href="#">1281</a> , <a href="#">1284</a> , <a href="#">1287</a> , <a href="#">1290</a> , <a href="#">1319</a> , <a href="#">1325</a> , <a href="#">1328</a> , <a href="#">1331</a> , <a href="#">1334</a> , <a href="#">1337</a>	\__postnotes_prop_item:nn .... <a href="#">4</a> , <a href="#">75</a> , <a href="#">80</a> , <a href="#">1098</a> , <a href="#">1103</a> , <a href="#">1110</a> , <a href="#">1185</a> , <a href="#">1257</a>
\l__postnotes_print_as_list_bool	.... <a href="#">227</a> , <a href="#">234</a> , <a href="#">239</a> , <a href="#">850</a> , <a href="#">917</a> , <a href="#">955</a> , <a href="#">978</a> , <a href="#">1776</a> , <a href="#">1791</a> , <a href="#">1798</a> , <a href="#">1806</a> , <a href="#">1816</a> , <a href="#">1841</a>	\g__postnotes_queue_seq .....
\l__postnotes_print_content_tl .	..... <a href="#">801</a> , <a href="#">895</a> , <a href="#">896</a> , <a href="#">935</a> , <a href="#">970</a>	.... <a href="#">18</a> , <a href="#">502</a> , <a href="#">535</a> , <a href="#">770</a> , <a href="#">832</a> , <a href="#">835</a> , <a href="#">1184</a>
\l__postnotes_print_counter_tl .	..... <a href="#">801</a> , <a href="#">932</a> , <a href="#">938</a>	\c__postnotes_ref_prefix_tl ....
\l__postnotes_print_env_tl .	..... <a href="#">226</a> , <a href="#">236</a> , <a href="#">240</a> , <a href="#">918</a> , <a href="#">980</a>	..... <a href="#">4</a> , <a href="#">84</a> , <a href="#">116</a> , <a href="#">187</a> , <a href="#">188</a> , <a href="#">194</a> , <a href="#">195</a> , <a href="#">551</a> , <a href="#">1095</a> , <a href="#">1131</a> , <a href="#">1132</a>
\l__postnotes_print_format_tl .	..... <a href="#">219</a> , <a href="#">222</a> , <a href="#">921</a>	\l__postnotes_restore_tmp_tl ...
\g__postnotes_print_labelseq_ queue_seq	<a href="#">828</a> , <a href="#">1014</a> , <a href="#">1051</a> , <a href="#">1078</a> , <a href="#">1135</a>	... <a href="#">1420</a> , <a href="#">1456</a> , <a href="#">1457</a> , <a href="#">1459</a> , <a href="#">1460</a> , <a href="#">1462</a> , <a href="#">1463</a> , <a href="#">1465</a> , <a href="#">1466</a> , <a href="#">1523</a> , <a href="#">1525</a> , <a href="#">1529</a> , <a href="#">1531</a> , <a href="#">1607</a> , <a href="#">1608</a> , <a href="#">1610</a> , <a href="#">1611</a> , <a href="#">1960</a> , <a href="#">1961</a> , <a href="#">1964</a> , <a href="#">1968</a> , <a href="#">1969</a> , <a href="#">1972</a>
\l__postnotes_print_mark_tl .	..... <a href="#">801</a> , <a href="#">929</a> , <a href="#">940</a> , <a href="#">951</a>	\l__postnotes_saved_spacefactor_ multi_tl .....
\l__postnotes_print_note_id_ next_tl	..... <a href="#">801</a> , <a href="#">865</a> , <a href="#">870</a> , <a href="#">872</a> , <a href="#">886</a> , <a href="#">888</a> , <a href="#">891</a> , <a href="#">905</a> , <a href="#">910</a> , <a href="#">912</a>	..... <a href="#">392</a> , <a href="#">418</a> , <a href="#">425</a>
\__postnotes_print_notes: .	..... <a href="#">26</a> , <a href="#">31</a> , <a href="#">35</a> , <a href="#">36</a> , <a href="#">794</a> , <a href="#">818</a> , <a href="#">818</a>	\l__postnotes_saved_spacefactor_ tl .....
\l__postnotes_print_plain_mark_ bool	..... <a href="#">20</a> , <a href="#">606</a> , <a href="#">612</a> , <a href="#">618</a> , <a href="#">1623</a> , <a href="#">1640</a> , <a href="#">1660</a> , <a href="#">1682</a>	..... <a href="#">705</a> , <a href="#">711</a> , <a href="#">721</a>
\l__postnotes_print_plain_mark_ stepcounter_bool	.... <a href="#">20</a> , <a href="#">50</a> , <a href="#">607</a> , <a href="#">613</a> , <a href="#">624</a> , <a href="#">1624</a> , <a href="#">1641</a> , <a href="#">1661</a> , <a href="#">1683</a>	\g__postnotes_sectid_int <a href="#">51</a> , <a href="#">764</a> , <a href="#">768</a>
\g__postnotes_print_postnotes_ int	. <a href="#">801</a> , <a href="#">821</a> , <a href="#">837</a> , <a href="#">1023</a> , <a href="#">1407</a> , <a href="#">1411</a>	\__postnotes_section:nn .....
\g__postnotes_print_queue_seq .	..... <a href="#">27</a> , <a href="#">33-35</a> , <a href="#">38</a> , <a href="#">801</a> , <a href="#">827</a> , <a href="#">831</a> , <a href="#">834</a> , <a href="#">838</a> , <a href="#">844</a> , <a href="#">845</a> , <a href="#">846</a> , <a href="#">848</a> , <a href="#">855</a> , <a href="#">857</a> , <a href="#">863</a> , <a href="#">869</a> , <a href="#">903</a> , <a href="#">909</a>	..... <a href="#">24</a> , <a href="#">761</a> , <a href="#">764</a> , <a href="#">765</a>
\l__postnotes_print_type_curr_tl	..... <a href="#">801</a> , <a href="#">860</a> , <a href="#">861</a> , <a href="#">899</a> , <a href="#">989</a>	\l__postnotes_section_exp_bool .
\l__postnotes_print_type_next_tl	.... <a href="#">801</a> , <a href="#">866</a> , <a href="#">873</a> , <a href="#">875</a> , <a href="#">906</a> , <a href="#">913</a> , <a href="#">975</a>	..... <a href="#">774</a> , <a href="#">784</a> , <a href="#">789</a>
\l__postnotes_print_type_prev_tl	..... <a href="#">801</a> , <a href="#">843</a> , <a href="#">898</a> , <a href="#">915</a> , <a href="#">988</a>	\g__postnotes_section_name_tl .
\l__postnotes_print_typeset_ mark_tl	..... <a href="#">801</a> , <a href="#">941</a> , <a href="#">960</a> , <a href="#">967</a>	..... <a href="#">49</a> , <a href="#">771</a> , <a href="#">783</a> , <a href="#">787</a>
\__postnotes_prop_gc_clear:n .	..... <a href="#">4</a> , <a href="#">75</a> , <a href="#">82</a> , <a href="#">995</a>	\__postnotes_set_babel_language:nn
\__postnotes_prop_get:nnN .	.... <a href="#">4</a> , <a href="#">75</a> , <a href="#">75</a> , <a href="#">859</a> , <a href="#">871</a> , <a href="#">879</a> , <a href="#">882</a> ,	..... <a href="#">62</a> , <a href="#">2124</a> , <a href="#">2124</a> , <a href="#">2145</a> , <a href="#">2146</a> , <a href="#">2147</a> , <a href="#">2148</a> , <a href="#">2149</a> , <a href="#">2150</a> , <a href="#">2151</a> , <a href="#">2152</a> , <a href="#">2161</a> , <a href="#">2162</a> , <a href="#">2163</a> , <a href="#">2164</a> , <a href="#">2173</a> , <a href="#">2174</a> , <a href="#">2183</a> , <a href="#">2184</a> , <a href="#">2185</a> , <a href="#">2186</a> , <a href="#">2187</a> , <a href="#">2188</a>

<code>\__postnotes_set_user_labels:</code> . . .	<code>\l__postnotes_tmpb_tl</code> . . . . .
. . . . . <a href="#">46</a> , <a href="#">560</a> , <a href="#">724</a> , <a href="#">724</a>	. . . . . <a href="#">16</a> ,
<code>\l__postnotes_sort_bool</code> <a href="#">430</a> , <a href="#">433</a> , <a href="#">1144</a>	<a href="#">1027</a> , <a href="#">1029</a> , <a href="#">1031</a> , <a href="#">1038</a> , <a href="#">1043</a> , <a href="#">1047</a> ,
<code>\l__postnotes_sort_num_fp</code> <a href="#">36</a> , <a href="#">579</a> , <a href="#">588</a>	<a href="#">1150</a> , <a href="#">1151</a> , <a href="#">1154</a> , <a href="#">1157</a> , <a href="#">1162</a> , <a href="#">1164</a>
<code>\__postnotes_sort_queue:N</code> . . . . .	<code>\__postnotes_typeset_mark:nnN</code> . . .
. . . . . <a href="#">35</a> , <a href="#">845</a> , <a href="#">1142</a> , <a href="#">1142</a>	. . . . . <a href="#">22</a> , <a href="#">569</a> , <a href="#">688</a> , <a href="#">688</a> , <a href="#">704</a>
<code>\__postnotes_split_labelseq:</code> . . .	<code>\__postnotes_typeset_mark_-</code>
. . . . . <a href="#">822</a> , <a href="#">1014</a> , <a href="#">1015</a>	<code>wrapper:nnn</code> . . . . .
<code>\__postnotes_step_counteraux:nnn</code>	. . . . . <a href="#">22</a> , <a href="#">643</a> , <a href="#">688</a> , <a href="#">690</a> , <a href="#">706</a> , <a href="#">743</a> , <a href="#">1560</a>
. . . . . <a href="#">4</a> , <a href="#">84</a> , <a href="#">100</a> , <a href="#">115</a>	<code>\__postnotes_typeset_text_-</code>
<code>\__postnotes_store:nn</code> . . <a href="#">3</a> , <a href="#">24</a> , <a href="#">25</a> , <a href="#">541</a>	<code>mark:nn</code> . . <a href="#">31</a> , <a href="#">949</a> , <a href="#">1001</a> , <a href="#">1001</a> , <a href="#">1013</a>
<code>\__postnotes_store_labelseq:nn</code> . .	<code>\l__postnotes_zrefhyperref_bool</code> .
. . . . . <a href="#">4</a> , <a href="#">84</a> , <a href="#">90</a> , <a href="#">114</a>	. . . . . <a href="#">1566</a> , <a href="#">1580</a> , <a href="#">1582</a>
<code>\__postnotes_store_section:nn</code> . . .	<code>\postnotessection</code> . . . . . <a href="#">3</a> , <a href="#">24</a> , <a href="#">25</a> , <a href="#">27</a> , <a href="#">758</a>
. . . . . <a href="#">3</a> , <a href="#">58</a> , <a href="#">58</a> , <a href="#">74</a> , <a href="#">777</a> , <a href="#">780</a>	<code>\postnotesetup</code> . . <a href="#">15</a> , <a href="#">465</a> , <a href="#">499</a> , <a href="#">1422</a> , <a href="#">1628</a>
<code>\l__postnotes_tabularx_inside_-</code>	<code>\postnotetext</code> . . . . . <a href="#">17</a>
<code>env_bool</code> . . . <a href="#">1647</a> , <a href="#">1650</a> , <a href="#">1656</a> , <a href="#">1668</a>	<code>\postnoterezref</code> . . . . . <a href="#">47</a> , <a href="#">52</a> , <a href="#">60</a> , <a href="#">1553</a>
<code>\__postnotes_tabularx_saved_-</code>	prg commands:
<code>write:Nn</code> . . . . . <a href="#">1651</a> , <a href="#">1657</a> , <a href="#">1669</a>	<code>\prg_new_protected_conditional:Npnn</code>
<code>\g__postnotes_tagsup_crossrefs_-</code>	. . . . . <a href="#">609</a>
<code>prop</code> . . . . . <a href="#">1982</a> , <a href="#">1985</a> , <a href="#">1995</a>	<code>\prg_return_false:</code> . . . . . <a href="#">650</a>
<code>\__postnotes_tagsup_gput_ref:nn</code> .	<code>\prg_return_true:</code> . . . . . <a href="#">649</a>
. . . . . <a href="#">59</a> , <a href="#">1988</a> , <a href="#">1988</a> , <a href="#">1997</a>	<code>\printpostnotes</code> . . . . .
<code>\__postnotes_tagsup_store_-</code>	. . . . . <a href="#">17</a> , <a href="#">18</a> , <a href="#">25–27</a> , <a href="#">30–33</a> , <a href="#">36</a> ,
<code>crossref:nN</code> . . . . .	<a href="#">37</a> , <a href="#">39</a> , <a href="#">40</a> , <a href="#">42</a> , <a href="#">44</a> , <a href="#">45</a> , <a href="#">52</a> , <a href="#">54</a> , <a href="#">59</a> , <a href="#">793</a>
. . . . . <a href="#">58</a> , <a href="#">1983</a> , <a href="#">2020</a> , <a href="#">2039</a> , <a href="#">2072</a> , <a href="#">2094</a>	prop commands:
<code>\__postnotes_tagsup_store_-</code>	<code>\prop_gc clear:N</code> . . . <a href="#">83</a> , <a href="#">1236</a> , <a href="#">1237</a> ,
<code>sstructnum:nN</code> . . . . .	<a href="#">1238</a> , <a href="#">1239</a> , <a href="#">1240</a> , <a href="#">1241</a> , <a href="#">1242</a> , <a href="#">1243</a>
. . . . . <a href="#">58</a> , <a href="#">1729</a> , <a href="#">1747</a> , <a href="#">1785</a> , <a href="#">1825</a> , <a href="#">1977</a>	<code>\prop_get:NnNTF</code> . . <a href="#">77</a> , <a href="#">1345</a> , <a href="#">1349</a> ,
<code>\__postnotes_tagsup_store_-</code>	<a href="#">1357</a> , <a href="#">1361</a> , <a href="#">1369</a> , <a href="#">1373</a> , <a href="#">1381</a> , <a href="#">1385</a>
<code>sstructnum:nN\__postnotes_-</code>	<code>\prop_gpop:NnNTF</code> . . . . . <a href="#">526</a>
<code>tagsup_store_crossref:nN</code> . . . <a href="#">1976</a>	<code>\prop_gput:Nnn</code> . . . . . <a href="#">28</a> , <a href="#">29</a> ,
<code>\g__postnotes_tagsup_structnums_-</code>	<a href="#">31</a> , <a href="#">33</a> , <a href="#">42</a> , <a href="#">46</a> , <a href="#">48</a> , <a href="#">50</a> , <a href="#">52</a> , <a href="#">54</a> , <a href="#">61</a> , <a href="#">65</a> ,
<code>prop</code> . . . . . <a href="#">1976</a> , <a href="#">1979</a> , <a href="#">1998</a>	<a href="#">69</a> , <a href="#">71</a> , <a href="#">107</a> , <a href="#">1280</a> , <a href="#">1283</a> , <a href="#">1286</a> , <a href="#">1289</a> ,
<code>\l__postnotes_tmpa_box</code> . . . . .	<a href="#">1293</a> , <a href="#">1298</a> , <a href="#">1303</a> , <a href="#">1308</a> , <a href="#">1327</a> , <a href="#">1330</a> ,
. . . . . <a href="#">16</a> , <a href="#">320</a> , <a href="#">322</a> , <a href="#">323</a>	<a href="#">1333</a> , <a href="#">1336</a> , <a href="#">1438</a> , <a href="#">1440</a> , <a href="#">1442</a> , <a href="#">1445</a> ,
<code>\l__postnotes_tmpa_seq</code> . . . . .	<a href="#">1599</a> , <a href="#">1601</a> , <a href="#">1940</a> , <a href="#">1951</a> , <a href="#">1979</a> , <a href="#">1985</a>
. . . . . <a href="#">16</a> , <a href="#">1018</a> , <a href="#">1046</a> , <a href="#">1052</a> , <a href="#">1053</a> , <a href="#">1055</a> ,	<code>\prop_item:Nn</code> . . . . . <a href="#">81</a> , <a href="#">1998</a>
<a href="#">1073</a> , <a href="#">1079</a> , <a href="#">1091</a> , <a href="#">1100</a> , <a href="#">1106</a> , <a href="#">1115</a> ,	<code>\prop_map_inline:Nn</code> . . . . . <a href="#">1995</a>
<a href="#">1128</a> , <a href="#">1135</a> , <a href="#">1180</a> , <a href="#">1184</a> , <a href="#">1187</a> , <a href="#">1190</a>	<code>\prop_new:N</code> . . . . .
<code>\l__postnotes_tmpa_tl</code> . . . . . <a href="#">16</a> ,	. . . . . <a href="#">27</a> , <a href="#">60</a> , <a href="#">87</a> , <a href="#">1207</a> , <a href="#">1208</a> , <a href="#">1209</a> , <a href="#">1210</a> ,
<a href="#">41</a> , <a href="#">43</a> , <a href="#">45</a> , <a href="#">47</a> , <a href="#">55</a> , <a href="#">64</a> , <a href="#">66</a> , <a href="#">68</a> , <a href="#">70</a> , <a href="#">72</a> ,	<a href="#">1211</a> , <a href="#">1212</a> , <a href="#">1213</a> , <a href="#">1214</a> , <a href="#">1976</a> , <a href="#">1982</a>
<a href="#">527</a> , <a href="#">528</a> , <a href="#">529</a> , <a href="#">632</a> , <a href="#">635</a> , <a href="#">776</a> , <a href="#">778</a> ,	property commands:
<a href="#">1022</a> , <a href="#">1024</a> , <a href="#">1025</a> , <a href="#">1027</a> , <a href="#">1149</a> , <a href="#">1151</a> ,	<code>\property_if_recorded:nnTF</code> . . . . .
<a href="#">1153</a> , <a href="#">1156</a> , <a href="#">1160</a> , <a href="#">1164</a> , <a href="#">1297</a> , <a href="#">1300</a> ,	. . . . . <a href="#">676</a> , <a href="#">2005</a> , <a href="#">2024</a> , <a href="#">2076</a>
<a href="#">1302</a> , <a href="#">1305</a> , <a href="#">1307</a> , <a href="#">1310</a> , <a href="#">1346</a> , <a href="#">1347</a> ,	<code>\property_new:nnnn</code> . . . . . <a href="#">735</a> , <a href="#">1927</a>
<a href="#">1350</a> , <a href="#">1352</a> , <a href="#">1354</a> , <a href="#">1358</a> , <a href="#">1359</a> , <a href="#">1362</a> ,	<code>\property_record:nn</code> . . . . .
<a href="#">1364</a> , <a href="#">1366</a> , <a href="#">1370</a> , <a href="#">1371</a> , <a href="#">1374</a> , <a href="#">1376</a> ,	. . . . . <a href="#">729</a> , <a href="#">1921</a> , <a href="#">1963</a> , <a href="#">1971</a>
<a href="#">1378</a> , <a href="#">1382</a> , <a href="#">1383</a> , <a href="#">1386</a> , <a href="#">1388</a> , <a href="#">1390</a> ,	<code>\property_ref:nn</code> <a href="#">682</a> , <a href="#">2011</a> , <a href="#">2030</a> , <a href="#">2082</a>
<a href="#">2009</a> , <a href="#">2018</a> , <a href="#">2021</a> , <a href="#">2028</a> , <a href="#">2037</a> , <a href="#">2040</a> ,	<code>\providecommand</code> . . . . .
<a href="#">2062</a> , <a href="#">2070</a> , <a href="#">2073</a> , <a href="#">2080</a> , <a href="#">2092</a> , <a href="#">2095</a>	. . . . . <a href="#">5</a> , <a href="#">124</a> , <a href="#">129</a> , <a href="#">134</a> , <a href="#">145</a> , <a href="#">150</a> , <a href="#">155</a>
<code>\l__postnotes_tmpb_seq</code> . . . . .	<code>\ProvidesExplPackage</code> . . . . . <a href="#">14</a>
. . . . . <a href="#">16</a> , <a href="#">1019</a> , <a href="#">1042</a> , <a href="#">1053</a> , <a href="#">1071</a> , <a href="#">1080</a>	

**R**

<code>\ref</code> . . . . .	<a href="#">2</a> , <a href="#">750</a> , <a href="#">752</a>
-----------------------------	---





\@ifl@t@r	5	\postnote@setcounteraux	5, 14, 130, 151, 475
\@mainaux	122, 127, 132, 143, 148, 153	\postnotes@required@kernel	3, 4, 6, 11
\@makecaption	43	\protected@edef	41, 45, 64, 68, 531, 538, 627, 632, 680, 776, 939
\@makefnmark	9	\protected@write	164, 484, 494
\@mkboth	209, 216	\z@	1506
\@newl@bel	4, 116	\zref@extract	667, 2064
\@textsuperscript	287, 321	\zref@extractdefault	1570
\blx@bibfiles	46	\zref@ifrefcontainsprop	661, 2059
\blx@edef@refcontext	1466, 1477	\zref@ifrefundefined	658
\blx@endrefsection	1535	\zref@localaddprop	1552, 1934
\blx@err@endnote	1430	\zref@newprop	1550, 1931
\blx@info	1534	\text	16, 17, 49
\blx@lasthash@foot	1493	\textnormal	49
\blx@lasthash@text	1492	\textsuperscript	12, 13
\blx@lastkey@foot	1491	\textup	49
\blx@lastkey@text	1490	\thechapter	36, 41, 64
\blx@lastmpfn	1506	\theHpostnote	17
\blx@refcontext@context	1446	\thepage	36, 171, 174
\blx@refcontext@labelalphaname	1476, 1483	\thepostnote	20, 531, 627, 632
\blx@refcontext@labelprefix	1471	\thesection	36, 45, 68
\blx@refcontext@labelprefix@real	1472	tl commands:	
\blx@refcontext@sortingnamekey	1474, 1480	\c_empty_tl	196
\blx@refcontext@sortingname	1473, 1479	\tl_clear:N	55, 72, 78, 189, 529, 1248, 1249, 1250, 1251, 1252, 1253, 1471, 1472, 1488, 1489, 1496, 1499, 1502, 1505
\blx@refcontext@uniquename	1475, 1482	\tl_const:Nn	84, 169, 390
\blx@sorting	1473	\tl_gclear:N	771, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1244, 1245, 1246, 1247
\blx@theendnote	1429	\tl_gput_right:Nn	62
\blx@theendnotetext	1430	\tl_gset:Nn	539, 1347, 1348, 1352, 1353, 1356, 1359, 1360, 1364, 1365, 1368, 1371, 1372, 1376, 1377, 1380, 1383, 1384, 1388, 1389, 1392, 1517, 2122
\blx@trackhash@foot	1497, 1499	\tl_gset_eq:NN	1530
\blx@trackhash@text	1494, 1496	\tl_if_empty:NTF	520, 622, 1262, 1278, 1325, 1961, 1969
\blx@trackkeys@foot	1503, 1505	\tl_if_eq:NNTF	1134, 1151, 1264, 1416, 1524
\blx@trackkeys@text	1500, 1502	\tl_if_eq:NnTF	861, 875, 915, 975
\c@blx@maxsection	1528	\tl_if_eq:nnTF	232, 1256
\c@page	169, 1402	\tl_if_eq_p:NN	1027
\c@postnote	20, 32, 37, 528, 631	\tl_item:Nn	1031, 1038, 1047
\c@postnotetext	937	\tl_item:nn	1061, 1068, 1074, 1181
\c@refsection	1439, 1457, 1488, 1489, 1516	\tl_new:N	16, 17, 219, 226, 293, 294, 295, 389, 392, 503, 507, 508, 577, 705, 783, 795, 796, 797, 798, 799, 800, 803, 804, 805, 806, 807, 808, 809, 810, 811, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206,
\c@refsegment	1441, 1460		
\c@zc@abschap	1600, 1608		
\c@zc@abssec	1602, 1611		
\FN@mf@check	12		
\FN@mf@prepare	12		
\hyper@linkend	697, 1009		
\hyper@linkstart	696, 1008		
\ifmeasuring@	48		
\p@postnote	538, 940		
\post@note	4, 5, 84, 125, 146, 166		
\postnote@addtcounteraux	5, 14, 135, 156, 475		



1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1420, 1511, 1512, 2111, 2112, 2113, 2114, 2121	146, 150, 151, 155, 156, 166, 485, 495
<code>\tl_set:Nn</code> . . . . . 188, 235, 236, 240, 418, 504, 537, 555, 635, 665, 711, 843, 865, 866, 885, 898, 905, 906, 936, 941, 988, 1022, 1319, 1473, 1474, 1475, 1476, 1515, 2009, 2028, 2062, 2080, 2115, 2116, 2117, 2118, 2140, 2141, 2142, 2143, 2156, 2157, 2158, 2159, 2168, 2169, 2170, 2171, 2178, 2179, 2180, 2181	<code>\topsep</code> . . . . . 258, 275, 1894, 1910
<code>\tl_set_eq:NN</code> . . . . . 542, 639	<b>U</b>
<code>\togglefalse</code> . . . . . 1470	<code>\undef</code> . . . . . 1490, 1491, 1492, 1493
<code>\toggletrue</code> . . . . . 1426	<code>\unkern</code> . . . . . 420, 421
token commands:	use commands:
<code>\token_to_str:N</code> . . . . . . 124, 125, 129, 130, 134, 135, 145,	<code>\use:N</code> . . . . . 1463
	<code>\use_none:n</code> . . . . . 1533, 1534
	<code>\UseHook</code> . . . . . 56, 534, 614, 842, 924
	<code>\UseInstance</code> . . . . . 1885, 1901
	<b>W</b>
	<code>\write</code> . . . . . 33, 1651, 1657, 1669
	<b>Z</b>
	<code>\zcsetup</code> . . . . . 1585, 1591
	<code>\zlabel</code> . . . . . 52, 57, 60, 733
	<code>\zref</code> . . . . . 1571, 1573