

# The `postnotes` package

## Code documentation

`gusbrs`

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

Version v0.5.1 – 2024-12-05

### 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>24</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>37</b>
<b>9</b>	<b>Compatibility</b>	<b>40</b>
<b>10</b>	<b>Languages</b>	<b>59</b>
	<b>Index</b>	<b>62</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} {2024-12-05} {0.5.1}
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_counter_aux_int
87 \prop_new:N \g__postnotes_counter_aux_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 { \seq_gput_right:Nn \g__postnotes_labelseq_seq { {#1} {#2} } }
92 \cs_new_protected:Npn \__postnotes_step_counter_aux:nnn #1#2#3
93 {
94   \bool_lazy_and:nnT
95     { \g__postnotes_counter_aux_bool }
96     { \str_if_eq_p:nn {#1} { mark } }
97   {
98     \int_gadd:Nn \g__postnotes_postnote_counter_aux_int { #3 }
99     \prop_gput:Nne \g__postnotes_counter_aux_prop { #2 }
100     { \int_use:N \g__postnotes_postnote_counter_aux_int }
101   }
102 }
103 \cs_new_protected:Npe \post@note #1#2#3#4
104 {
105   \exp_not:N \bool_gset_false:N \exp_not:N \g__postnotes_firststrun_bool
106   \exp_not:N \__postnotes_store_labelseq:nn { #1 } { #2 }
107   \exp_not:N \__postnotes_step_counter_aux:nnn { #1 } { #2 } { #4 }
108   \exp_not:N \@newl@bel { \c__postnotes_ref_prefix_tl } { #1 @ #2 } { #3 }
109 }

```

(End of definition for \post@note, \\_\_postnotes\_store\_labelseq:nn, and \\_\_postnotes\_step\_counter\_aux: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.

```

110 \AddToHook { begindocument }
111 {
112   \legacy_if:nT { @filesw }
113   {
114     \iow_now:Ne \@mainaux
115     {
116       \token_to_str:N \providecommand
117       \token_to_str:N \post@note [4] { }
118     }
119     \iow_now:Ne \@mainaux
120     {
121       \token_to_str:N \providecommand
122       \token_to_str:N \postnote@setcounteraux [1] { }
123     }
124     \iow_now:Ne \@mainaux
125     {
126       \token_to_str:N \providecommand
127       \token_to_str:N \postnote@addtocounteraux [1] { }
128     }
129   }
130 }
131 \AddToHook { include/before }
132 {

```

```

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

```

`\__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_print_label:n

```

```

\__postnotes_set_label:n {<label type>} {<note id>} {<value>}
  {<counteraux step>}
\__postnotes_set_mark_page_label:n {<note id>} {<counteraux step>}
\__postnotes_set_section_page_label:n {<note id>}
\__postnotes_set_print_label:n {<note id>}

```

```

152 \cs_new_protected:Npn \__postnotes_set_label:n #1#2#3#4
153 {
154   \legacy_if:nT { @filesw }
155   {
156     \protected@write \@auxout { }
157     { \token_to_str:N \post@note { #1 } { #2 } { #3 } { #4 } }
158   }
159 }
160 \cs_new_protected:Npn \__postnotes_set_mark_page_label:n #1#2
161 { \__postnotes_set_label:n { mark } { #1 } { \thepage } { #2 } }
162 \cs_generate_variant:Nn \__postnotes_set_mark_page_label:n { ee }
163 \cs_new_protected:Npn \__postnotes_set_section_page_label:n #1
164 { \__postnotes_set_label:n { section } { #1 } { \thepage } { } }
165 \cs_generate_variant:Nn \__postnotes_set_section_page_label:n { e }
166 \cs_new_protected:Npn \__postnotes_set_print_label:n #1
167 { \__postnotes_set_label:n { print } { #1 } { } { } }
168 \cs_generate_variant:Nn \__postnotes_set_print_label:n { e }

```

*(End of definition for `\__postnotes_set_label:n` 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>}

```

```

169 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
170 {
171   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
172     { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
173     { \tl_clear:N #1 }
174 }
175 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Ne }
176 \cs_new:Npn \__postnotes_extract_pageref:n #1
177 {
178   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
179     { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
180     { \c_empty_tl }
181 }
182 \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

```

183 \keys_define:nn { postnotes/setup }
184 {
185   heading .cs_set_protected:Np = \pnheading ,
186   heading .value_required:n = true ,
187 }

```

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

```

188 \cs_if_exist:cTF { chapter }
189 {
190   \cs_new_protected:Npn \pnheading
191     {
192       \chapter*{\pntitle}
193       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
194     }
195 }
196 {
197   \cs_new_protected:Npn \pnheading
198     {
199       \section*{\pntitle}
200       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
201     }
202 }

```

*(End of definition for \pnheading.)*

`\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`

```

203 \NewDocumentCommand \pnheaderdefault {}
204 {
205   \group_begin:
206   \protected@edef \l__postnotes_tmpa_tl { \FirstMark{postnotes/page} }
207   \protected@edef \l__postnotes_tmpb_tl { \LastMark{postnotes/page} }
208   \tl_if_eq:NNTF \l__postnotes_tmpa_tl \l__postnotes_tmpb_tl
209     { \pnhdnotes{ }~\pnhdtopage{ }~ \FirstMark{postnotes/page} }
210     {
211       \pnhdnotes{ }~\pnhdtopages{ }~
212       \FirstMark{postnotes/page}--\LastMark{postnotes/page}
213     }
214   \group_end:
215 }

```

(End of definition for \pnheaderdefault.)

### format option

```

216 \tl_new:N \l__postnotes_print_format_tl
217 \keys_define:nn { postnotes/setup }
218 {
219   format .tl_set:N = \l__postnotes_print_format_tl ,
220   format .initial:n = { \small } ,
221   format .value_required:n = true ,
222 }

```

### listenv option

```

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

```

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

```

233         \tl_set:Nn \l__postnotes_print_env_tl { itemize }
234       }
235     {
236       \bool_set_true:N \l__postnotes_print_as_list_bool
237       \tl_set:Nn \l__postnotes_print_env_tl {#1}
238     }
239   } ,
240   listenv .initial:n = { postnoteslist } ,
241   listenv .value_required:n = true ,
242 }

```

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*).

```

243 \NewDocumentEnvironment { postnoteslist } { }

```



```

244 {
245   \list { }
246   {
247     \setlength { \leftmargin } { Opt }
248     \setlength { \labelwidth } { Opt }
249     \setlength { \itemindent } { .5\parindent }
250     \cs_set_eq:NN \makelabel \_postnotes_list_makelabel:n
251     \setlength { \rightmargin } { Opt }
252     \setlength { \listparindent } { \parindent }
253     \setlength { \parsep } { \parskip }
254     \setlength { \itemsep } { Opt }
255     \setlength { \topsep } { .5\topsep }
256     \setlength { \partopsep } { .5\partopsep }
257   }
258 }
259 { \endlist }
260 \NewDocumentEnvironment { postnoteslisthang } { }
261 {
262   \list { }
263   {
264     \setlength { \leftmargin } { 1em }
265     \setlength { \labelwidth } { -\leftmargin }
266     \setlength { \itemindent } { -2\leftmargin }
267     \cs_set_eq:NN \makelabel \_postnotes_list_makelabel:n
268     \setlength { \rightmargin } { Opt }
269     \setlength { \listparindent } { \parindent }
270     \setlength { \parsep } { \parskip }
271     \setlength { \itemsep } { Opt }
272     \setlength { \topsep } { .5\topsep }
273     \setlength { \partopsep } { .5\partopsep }
274   }
275 }
276 { \endlist }
277 \cs_new:Npn \_postnotes_list_makelabel:n #1
278 { \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.

```

279 \keys_define:nn { postnotes/setup }
280 {
281   makemark .cs_set:Np = \_postnotes_make_mark:nnn #1#2#3 ,
282   makemark .value_required:n = true ,

```

From the default kernel definition of \@makefnmark.

```

283   makemark .initial:n =
284     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
285   maketextmark .cs_set:Np = \_postnotes_make_text_mark:nnn #1#2#3 ,
286   maketextmark .value_required:n = true ,
287   maketextmark .initial:n = { #2 #1 . #3 } ,
288 }
289 \cs_generate_variant:Nn \_postnotes_make_mark:nnn { Vnn }

```

## pretextmark, posttextmark, postprintnote options

```
290 \tl_new:N \l__postnotes_pre_textmark_tl
291 \tl_new:N \l__postnotes_post_textmark_tl
292 \tl_new:N \l__postnotes_post_printnote_tl
293 \keys_define:nn { postnotes/setup }
294 {
295   pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
296   pretextmark .value_required:n = true ,
297   posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
298   posttextmark .value_required:n = true ,
299   postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
300   postprintnote .value_required:n = true ,
301 }
```

## style option

```
302 \keys_define:nn { postnotes/setup }
303 {
304   style .choice: ,
305   style / endnotes .meta:n =
306   {
307     listenv = none ,
308     format =
309     {
310       \footnotesize
311       \setlength { \rightskip } { Opt }
312       \setlength { \leftskip } { Opt }
313       \setlength { \parindent } { 1.8em }
314     } ,
```

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.

```
315     maketextmark =
316     {
317       \hbox_set:Nn \l__postnotes_tmpa_box
318       { \@textsuperscript { \normalfont ##1 } }
319       \skip_horizontal:n { - \box_wd:N \l__postnotes_tmpa_box }
320       ##2 \box_use:N \l__postnotes_tmpa_box ##3
321     } ,
322   } ,
323   style / pagenote .meta:n =
324   {
325     listenv = none ,
326     format = { } ,
327     pretextmark = { \noindent } ,
328     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
329     posttextmark = { ~ } ,
330   } ,
331 }
```

## hyperref and backlink options

```
332 \bool_new:N \l__postnotes_hyperlink_bool
333 \bool_new:N \l__postnotes_hyperref_warn_bool
334 \bool_new:N \l__postnotes_backlink_bool
```

```

335 \keys_define:nn { postnotes/setup }
336 {
337   hyperref .choice: ,
338   hyperref / auto .code:n =
339     {
340       \bool_set_true:N \l__postnotes_hyperlink_bool
341       \bool_set_false:N \l__postnotes_hyperref_warn_bool
342     } ,
343   hyperref / true .code:n =
344     {
345       \bool_set_true:N \l__postnotes_hyperlink_bool
346       \bool_set_true:N \l__postnotes_hyperref_warn_bool
347     } ,
348   hyperref / false .code:n =
349     {
350       \bool_set_false:N \l__postnotes_hyperlink_bool
351       \bool_set_false:N \l__postnotes_hyperref_warn_bool
352     } ,
353   hyperref .initial:n = auto ,
354   hyperref .default:n = true ,
355   backlink .bool_set:N = \l__postnotes_backlink_bool ,
356   backlink .initial:n = true ,
357   backlink .default:n = true ,
358 }
359 \AddToHook { begindocument }
360 {
361   \IfPackageLoadedTF { hyperref }
362   {
363     {
364       \bool_if:NT \l__postnotes_hyperref_warn_bool
365       { \msg_warning:nn { postnotes } { missing-hyperref } }
366       \bool_set_false:N \l__postnotes_hyperlink_bool
367     }
368   \keys_define:nn { postnotes/setup }
369   {
370     hyperref .code:n =
371     {
372       \msg_warning:nnn { postnotes }
373       { option-preamble-only } { hyperref }
374     } ,
375     backlink .code:n =
376     {
377       \msg_warning:nnn { postnotes }
378       { option-preamble-only } { backlink }
379     } ,
380   }
381 }
382 \msg_new:nnn { postnotes } { option-preamble-only }
383 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
384 \msg_new:nnn { postnotes } { missing-hyperref }
385 { 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 `\multifootsep`. With the intention, naturally, that different classes and packages can “interwork” with regard to this feature. And this became a sort of “shared feature” in the ecosystem (the list includes `footmisc`, `KOMA-Script`, `eledmac`, `reledmac`, `tufte`, `memoir`, `parnotes`, `sidenotes`, and now also `postnotes`, see discussion at <https://chat.stackexchange.com/transcript/message/66421777#66421777>). What is crucial for this interplay, however, is not quite the variables themselves, but *the value of the canceling pair of kerns*, stored in `\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 `\multifootsep` though. `footmisc` stores a plain comma in `\multifootsep` and applies formatting around it in `\FN@mf@check` (hard-coded as `\textsuperscript`). `KOMA-Script` also stores plain content in `\multifootsep` and applies the formatting in the check function. `memoir`, however, stores the formatting directly in `\multifootsep` and applies no formatting later. Also, `latex-lab-footmisc.ltx`, in adding PDF tagging support for `footmisc` stores the tagging code directly in `\multifootsep`. I don’t know if there are others, but these cases are already relevant enough to spoil things. The problem is not that they disagree on the value of `\multifootsep`, but on the function(s) this macro is supposed to perform. That given, any other parties trying to partake in this feature have to handle things differently depending on the value of `\multifootsep`. All in all, `postnotes` takes the cautious stance of using internal variables, instead of the shared ones, to implement the `multiple` option. The only thing that really needs to be common is the value of the canceling kerns of `3sp` in `\_postnotes_multiple_prepare:`.

386 \tl\_new:N \l\_\_postnotes\_multisep\_tl

```

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

```

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`.

```

399 \cs_new_protected:Npn \__postnotes_multiple_prepare:
400 {
401   \bool_if:NT \l__postnotes_multiple_bool
402   {
403     \kern -\c_postnotes_multi_notemarker_tl
404     \kern \c_postnotes_multi_notemarker_tl
405     \scan_stop:
406   }
407 }

```

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).

```

408 \cs_new_protected:Npn \__postnotes_multiple_check:
409 {
410   \bool_if:NT \l__postnotes_multiple_bool
411   {
412     \dim_compare:nNnT
413     { \c_postnotes_multi_notemarker_tl } = { \lastkern }
414     {
415       \tl_set:Ne \l__postnotes_saved_spacefactor_multi_tl
416       { \int_use:N \spacefactor }
417       \unkern
418       \unkern
419       \tag_socket_use:n { postnotes/multisep/begin }
420       \__postnotes_make_mark:Vnn \l__postnotes_multisep_tl { } { }
421       \tag_socket_use:n { postnotes/multisep/end }
422       \spacefactor \l__postnotes_saved_spacefactor_multi_tl
423       \scan_stop:
424     }
425   }
426 }

```

## sort option

```

427 \bool_new:N \l__postnotes_sort_bool
428 \keys_define:nn { postnotes/setup }
429 {

```

```

430     sort .bool_set:N = \l__postnotes_sort_bool ,
431     sort .initial:n = true ,
432     sort .default:n = true ,
433 }

```

### checkduplicates and checkfloats options

```

434 \bool_new:N \l__postnotes_check_dupli_bool
435 \bool_new:N \l__postnotes_check_floats_bool
436 \keys_define:nn { postnotes/setup }
437 {
438     checkduplicates .bool_set:N = \l__postnotes_check_dupli_bool ,
439     checkduplicates .default:n = true ,
440     checkduplicates .initial:n = true ,
441     checkfloats .bool_set:N = \l__postnotes_check_floats_bool ,
442     checkfloats .default:n = true ,
443     checkfloats .initial:n = false ,
444 }

```

### maybemulti option

```

445 \bool_new:N \l__postnotes_maybe_multi_bool
446 \keys_define:nn { postnotes/setup }
447 {
448     maybemulti .bool_set:N = \l__postnotes_maybe_multi_bool ,
449     maybemulti .default:n = true ,
450     maybemulti .initial:n = false ,
451 }

```

### counteraux option

```

452 \bool_new:N \g__postnotes_counteraux_bool
453 \keys_define:nn { postnotes/setup }
454 {
455     counteraux .bool_gset:N = \g__postnotes_counteraux_bool ,
456     counteraux .default:n = true ,
457     counteraux .initial:n = false ,
458 }
459 \AddToHook { begindocument/before }
460 {
461     \bool_if:NT \g__postnotes_counteraux_bool
462     { \postnotessetup { sort=false } }
463     \keys_define:nn { postnotes/setup }
464     {
465         counteraux .code:n =
466         {
467             \msg_warning:nnn { postnotes }
468             { option-preamble-only } { counteraux }
469         } ,
470     }
471 }

```

```

\pnsetcounteraux{<int>}
\pnaddtounteraux{<int>}
\postnote@setcounteraux{<int>}
\postnote@addtounteraux{<int>}

```

```

\pnsetcounteraux
\pnaddtounteraux
\postnote@setcounteraux
\postnote@addtounteraux

```

```

472 \cs_new_protected:Npn \postnote@setcounteraux #1
473   { \int_gset:Nn \g__postnotes_postnote_counteraux_int { #1 } }
474 \cs_new_protected:Npn \postnote@addtounteraux #1
475   { \int_gadd:Nn \g__postnotes_postnote_counteraux_int { #1 } }
476 \NewDocumentCommand \pnsetcounteraux { m }
477   {
478     \@bsphack
479     \legacy_if:nT { @filesw }
480     {
481       \protected@write \@auxout { }
482         { \token_to_str:N \postnote@setcounteraux { #1 } }
483     }
484     \@esphack
485   }
486 \NewDocumentCommand \pnaddtounteraux { m }
487   {
488     \@bsphack
489     \legacy_if:nT { @filesw }
490     {
491       \protected@write \@auxout { }
492         { \token_to_str:N \postnote@addtounteraux { #1 } }
493     }
494     \@esphack
495   }

```

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

## \postnotesetup

\postnotesetup Provide \postnotesetup.

```

\postnotesetup{<options>}

496 \NewDocumentCommand \postnotesetup { m }
497   { \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.

```
498 \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 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`.

```
499 \int_new:N \g__postnotes_note_id_int
500 \tl_new:N \l_postnotes_note_id_tl
501 \tl_set:Nn \l_postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
502 \seq_new:N \g__postnotes_queue_seq
503 \int_new:N \l_postnotes_counteraux_step_int
504 \tl_new:N \l_postnotes_mark_typeset_tl
505 \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}
```

```
506 \NewDocumentCommand \postnote { 0 { } +m }
507 { \__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}
```

```
508 \NewHook { postnotes/note/begin }
509 \NewHook { postnotes/note/setlabels }
510 \cs_new_protected:Npn \__postnotes_note:nn #1#2
511 {
512   \group_begin:
513   \keys_set:nn { postnotes/note } {#1}
514   \bool_if:NT \l__postnotes_nomark_bool { \@bsphack }
515   \__postnotes_inhibit_note:F
516   {
517     \int_gincr:N \g__postnotes_note_id_int
518     \tl_if_empty:NTF \l__postnotes_mark_tl
519     {
520       \stepcounter { postnote }
521       \int_set:Nn \l_postnotes_counteraux_step_int { 1 }
522       \bool_if:NT \g__postnotes_counteraux_bool
523       {
524         \exp_args:NNe \prop_gpop:NnNT \g__postnotes_counteraux_prop
```

```

525         { \l_postnotes_note_id_tl } \l__postnotes_tmpa_tl
526         { \int_set:Nn \c@postnote { \l__postnotes_tmpa_tl } }
527         \tl_clear:N \l__postnotes_tmpa_tl
528     }
529     \protected@edef \l__postnotes_mark_tl { \thepostnote }
530 }
531 { \int_set:Nn \l__postnotes_counteraux_step_int { 0 } }
532 \UseHook { postnotes/note/begin }
533 \seq_gput_right:Ne \g__postnotes_queue_seq
534 { \l_postnotes_note_id_tl }
535 \tl_set:Nn \@currentcounter { postnote }
536 \protected@edef \@currentlabel { \p@postnote \l__postnotes_mark_tl }
537 \tl_gset:Ne \@currentHref
538 { postnote. \l_postnotes_note_id_tl .mark }
539 \__postnotes_store:nn { \l_postnotes_note_id_tl } {#2}
540 \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`.

```

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

```

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

Options for `\postnote`.

```

576 \tl_new:N \l__postnotes_mark_tl
577 \bool_new:N \l__postnotes_nomark_bool
578 \fp_new:N \l__postnotes_sort_num_fp
579 \str_new:N \l__postnotes_note_label_str
580 \bool_new:N \l__postnotes_manual_sortnum_bool
581 \keys_define:nn { postnotes/note }
582 {
583   markstr .tl_set:N = \l__postnotes_mark_tl ,
584   markstr .value_required:n = true ,
585   sortnum .code:n =
586     {
587       \fp_set:Nn \l__postnotes_sort_num_fp {#1}
588       \bool_set_true:N \l__postnotes_manual_sortnum_bool
589     } ,
590   sortnum .value_required:n = true ,
591   mark .meta:n =
592     {
593       markstr = {#1} ,
594       sortnum = {#1} ,
595     } ,
596   mark .value_required:n = true ,
597   nomark .bool_set:N = \l__postnotes_nomark_bool ,
598   nomark .default:n = true ,
599   label .str_set:N = \l__postnotes_note_label_str ,
600   label .value_required:n = true ,
601   maybemulti .bool_set:N = \l__postnotes_maybe_multi_bool ,
602   maybemulti .default:n = true ,
603 }

```

`\_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.

```

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

```

614     \bool_lazy_all:nT
615     {
616       { \l__postnotes_inhibit_note_bool }
617       { \l__postnotes_print_plain_mark_bool }
618       { ! \l__postnotes_nomark_bool }
619     }
620     {
621       \tl_if_empty:NTF \l__postnotes_mark_tl
622       {
623         \bool_if:NTF \l__postnotes_print_plain_mark_stepcounter_bool
624         {
625           \stepcounter { postnote }
626           \protected@edef \l__postnotes_mark_typeset_tl { \thepostnote }
627         }
628         {
629           \group_begin:
630           \int_incr:N \c@postnote
631           \protected@edef \l__postnotes_tmpa_tl { \thepostnote }
632           \exp_args:NNNV
633           \group_end:
634           \tl_set:Nn \l__postnotes_mark_typeset_tl \l__postnotes_tmpa_tl
635         }

```

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.

```

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

```

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

```

\_postnotes_get_label_if_exist:N      \_postnotes_get_label_if_exist:N <\l__postnotes_mark_tl>
651 \cs_new_protected:Npn \_postnotes_get_label_if_exist:N #1
652 {
653   \str_if_empty:NTF \l__postnotes_note_label_str
654   {
655     \str_if_empty:NF \l__postnotes_note_zlabel_str
656     {
657       \exp_args:NV \zref@ifrefundefined \l__postnotes_note_zlabel_str
658       { }
659       {
660         \exp_args:NV \zref@ifrefcontainsprop
661         \l__postnotes_note_zlabel_str
662         { postnote@mark }
663         {
664           \exp_args:NNNo \exp_args:NNNo \tl_set:Nn #1
665           {
666             \zref@extract
667             { \l__postnotes_note_zlabel_str } { postnote@mark }
668           }
669         }
670       } { }
671     }
672   }
673 }
674 {
675   \exp_args:Ne \property_if_recorded:nnT
676   { postnotes@ \l__postnotes_note_label_str }
677   { postnotes/mark }
678   {
679     \protected@edef #1
680     {
681       \property_ref:ee
682       { \_postnotes_ \l__postnotes_note_label_str } { postnotes/mark }
683     }
684   }
685 }
686 }

```

(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>}
687 \cs_new_protected:Npn \_postnotes_typeset_mark:nnN #1#2#3
688 {
689   \_postnotes_typeset_mark_wrapper:nnn
690   {

```

```

691     #3
692     \bool_if:NTF \l__postnotes_hyperlink_bool
693     {
694         \__postnotes_make_mark:nnn {#2}
695         { \hyper@linkstart { link } { postnote. #1 .text } }
696         { \hyper@linkend }
697     }
698     { \__postnotes_make_mark:nnn {#2} { } { } }
699 }
700 { \tag_socket_use:n { postnotes/mark/begin } }
701 { \tag_socket_use:n { postnotes/mark/end } }
702 }
703 \cs_generate_variant:Nn \__postnotes_typeset_mark:nnN { eVN }
704 \tl_new:N \l__postnotes_saved_spacefactor_tl
705 \cs_new_protected:Npn \__postnotes_typeset_mark_wrapper:nnn #1#2#3
706 {
707     \mode_leave_vertical:
708     \mode_if_horizontal:T
709     {
710         \tl_set:Nc \l__postnotes_saved_spacefactor_tl
711         { \int_use:N \spacefactor }
712         \__postnotes_multiple_check:
713         \nobreak
714     }
715     #2 % begin tagging socket
716     #1 % mark
717     #3 % end tagging socket
718     \__postnotes_multiple_prepare:
719     \mode_if_horizontal:T
720     { \spacefactor \l__postnotes_saved_spacefactor_tl }
721     \scan_stop:
722 }

```

*(End of definition for \\_\_postnotes\_typeset\_mark:nnN and \\_\_postnotes\_typeset\_mark\_wrapper:nnn.)*

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

```

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

```

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

## 5 \postnoteref

\postnoteref Provide \postnoteref.

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

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

## 6 \postnotesection

\postnotesection Provide \postnotesection.

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

(End of definition for \postnotesection.)

\\_\_postnotes\_section:nn The internal version of \postnotesection.



```

    \_postnotes_section:nn {<options>} {<content>}
763 \int_new:N \g__postnotes_sectid_int
764 \cs_new_protected:Npn \_postnotes_section:nn #1#2
765 {
766   \group_begin:
767     \int_gincr:N \g__postnotes_sectid_int
768     \int_gincr:N \g__postnotes_note_id_int
769     \seq_gput_right:Ne \g__postnotes_queue_seq { \l_postnotes_note_id_tl }
770     \tl_gclear:N \g__postnotes_section_name_tl
771     \keys_set:nn { postnotes/section } {#1}
772     \_postnotes_set_section_page_label:e { \l_postnotes_note_id_tl }
773     \bool_if:NTF \l__postnotes_section_exp_bool
774     {
775       \protected@edef \l__postnotes_tmpa_tl {#2}
776       \_postnotes_store_section:nV { \l_postnotes_note_id_tl }
777       \l__postnotes_tmpa_tl
778     }
779     { \_postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
780   \group_end:
781 }

```

(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.

```

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

```

## 7 `\printpostnotes`

`\printpostnotes` Provide `\printpostnotes`.

```

    \printpostnotes
792 \NewDocumentCommand \printpostnotes { }
793 { \_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>	794 <code>\tl_new:N \pnthechapter</code>
<code>\pnthesectionnextnote</code>	795 <code>\tl_new:N \pnthesection</code>
<code>\pnthepage</code>	
<code>\pnidnextnote</code>	

```

796 \tl_new:N \pnidnextnote
797 \tl_new:N \pnthechapternextnote
798 \tl_new:N \pnthesectionnextnote
799 \tl_new:N \pnthepage

```

(End of definition for \pnthechapter and others.)

Auxiliary variables for \\_\_postnotes\_print\_notes:.

```

\g__postnotes_print_postnotes_int
\g__postnotes_print_queue_seq
\l__postnotes_print_note_id_tl
\l__postnotes_print_note_id_next_tl
\l__postnotes_print_counter_tl
\l__postnotes_print_mark_tl
\l__postnotes_print_typeset_mark_tl
\l__postnotes_print_type_curr_tl
\l__postnotes_print_type_next_tl
\l__postnotes_print_type_prev_tl
\l__postnotes_print_content_tl
\l__postnotes_print_page_tl
\l__postnotes_print_chapter_tl
\l__postnotes_print_section_tl
\l__postnotes_print_sectname_tl
\l__postnotes_clear_queue_seq

```

```

800 \int_new:N \g__postnotes_print_postnotes_int
801 \seq_new:N \g__postnotes_print_queue_seq
802 \tl_new:N \l__postnotes_print_note_id_tl
803 \tl_new:N \l__postnotes_print_note_id_next_tl
804 \tl_new:N \l__postnotes_print_counter_tl
805 \tl_new:N \l__postnotes_print_mark_tl
806 \tl_new:N \l__postnotes_print_typeset_mark_tl
807 \tl_new:N \l__postnotes_print_type_curr_tl
808 \tl_new:N \l__postnotes_print_type_next_tl
809 \tl_new:N \l__postnotes_print_type_prev_tl
810 \tl_new:N \l__postnotes_print_content_tl
811 \tl_new:N \l__postnotes_print_page_tl
812 \tl_new:N \l__postnotes_print_chapter_tl
813 \tl_new:N \l__postnotes_print_section_tl
814 \tl_new:N \l__postnotes_print_sectname_tl
815 \seq_new:N \l__postnotes_clear_queue_seq

```

(End of definition for \g\_\_postnotes\_print\_postnotes\_int and others.)

ltmarks mark classes for running header support data.

```

816 \mark_new_class:n { postnotes/page }
817 \mark_new_class:n { postnotes/chapter }
818 \mark_new_class:n { postnotes/section }
819 \mark_new_class:n { postnotes/sectname }

```

\\_\_postnotes\_print\_notes:’ hooks. Meant to provide 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.

```

820 \NewHook { postnotes/print/begin }
821 \NewHook { postnotes/print/note/begin }
822 \NewHook { postnotes/print/note/typesetmark }
823 \NewHook { postnotes/print/note/ltmarks }

```

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 postnotessection 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 postnotessection 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.

```
824 \newcounter { postnotetext }
825 \newcounter { postnotesection }
826 \setcounter { postnotesection } { 10000 }
```

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

```

\__postnotes_print_notes:
827 \cs_new_protected:Npn \__postnotes_print_notes:
828 {
829   \group_begin:
830   \int_gincr:N \g__postnotes_print_postnotes_int
831   \__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`.

```
832   \bool_lazy_and:nnTF
833     { \g__postnotes_countersaux_bool }
834     { ! \g__postnotes_firstrun_bool }
835     {
836       \seq_gset_eq:NN \g__postnotes_print_queue_seq
837       \g__postnotes_print_labelseq_queue_seq
838     }
839     {
840       \seq_gset_eq:NN \g__postnotes_print_queue_seq
841       \g__postnotes_queue_seq
842     }
843   \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_print_queue_seq
844   \seq_gclear:N \g__postnotes_queue_seq
```

The purpose of this label is to provide a point for splitting the `labelseq` with the `counteraux` option. It only needs to exist, it doesn’t even store the page value.

```
845   \__postnotes_set_print_label:e
846     { \int_use:N \g__postnotes_print_postnotes_int }
847   \seq_if_empty:NTF \g__postnotes_print_queue_seq
848     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
849     {
850       \pnheading
851       \UseHook { postnotes/print/begin }
852       \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
853       \__postnotes_check_duplicates:N \g__postnotes_print_queue_seq
```

```

854     \_postnotes_sort_queue:N \g__postnotes_print_queue_seq
855     \_postnotes_check_floats:N \g__postnotes_print_queue_seq
856     \_postnotes_set_header_vars_first:

```

Ensure the first note after a heading has paragraph indentation when `listenv` is none. `endnotes` uses a workaroundsish solution in `\noteheading`, setting a box and then skipping back a line. Enrico Gregorio is correct, though, in criticizing it at [https://tex.stackexchange.com/q/575905#comment1450213\\_575915](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).

```

857     \bool_if:NF \l__postnotes_print_as_list_bool
858     {
859         \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
860         \@afterindenttrue
861     }
862 \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_print_queue_seq }
863 {
864     \seq_gpop_left:NN \g__postnotes_print_queue_seq
865     \l_postnotes_print_note_id_tl
866     \_postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
867     { type } \l__postnotes_print_type_curr_tl
868     \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
869     { % type_curr = 'section'
870         \seq_if_empty:NTF \g__postnotes_print_queue_seq
871         {
872             \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
873             \tl_set:Nn \l__postnotes_print_type_next_tl { close }
874         }
875         {
876             \seq_get_left:NN \g__postnotes_print_queue_seq
877             \l__postnotes_print_note_id_next_tl
878             \_postnotes_prop_get:nnN
879             { \l__postnotes_print_note_id_next_tl }
880             { type } \l__postnotes_print_type_next_tl
881         }

```

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

```

882     \tl_if_eq:NnT \l__postnotes_print_type_next_tl { note }
883     {
884         \stepcounter { postnotessection }
885         \group_begin:
886             \_postnotes_prop_get:nnN
887             { \l_postnotes_print_note_id_tl }
888             { thechapter } \pnthechapter
889             \_postnotes_prop_get:nnN
890             { \l_postnotes_print_note_id_tl }
891             { thesection } \pnthesection
892             \tl_set:NV \pnidnextnote
893             \l__postnotes_print_note_id_next_tl
894             \_postnotes_prop_get:nnN
895             { \l__postnotes_print_note_id_next_tl }
896             { thechapter } \pnthechapternextnote
897             \_postnotes_prop_get:nnN
898             { \l__postnotes_print_note_id_next_tl }
899             { thesection } \pnthesectionnextnote
900             \_postnotes_prop_get:nnN

```

```

901         { \l_postnotes_print_note_id_tl }
902         { content } \l__postnotes_print_content_tl
903         \l__postnotes_print_content_tl
904     \group_end:

```

Set `type_prev` for the next iteration.

```

905         \tl_set:NV \l__postnotes_print_type_prev_tl
906         \l__postnotes_print_type_curr_tl
907     }
908 }
909 { % type_curr = 'note'
910   \seq_if_empty:NTF \g__postnotes_print_queue_seq
911   {
912     \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
913     \tl_set:Nn \l__postnotes_print_type_next_tl { close }
914   }
915   {
916     \seq_get_left:NN \g__postnotes_print_queue_seq
917     \l__postnotes_print_note_id_next_tl
918     \__postnotes_prop_get:nnN
919     { \l__postnotes_print_note_id_next_tl }
920     { type } \l__postnotes_print_type_next_tl
921   }
922   \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }
923   {
924     \bool_if:NTF \l__postnotes_print_as_list_bool
925     { \exp_args:NV \begin \l__postnotes_print_env_tl }
926     { \group_begin: }
927     \tag_socket_use:n { postnotes/printlist/begin }
928     \l__postnotes_print_format_tl
929   }
930   \group_begin:
931   \UseHook { postnotes/print/note/begin }
932   \__postnotes_get_pageref:Ne \l__postnotes_print_page_tl
933   { mark@ \l__postnotes_print_note_id_tl }
934   \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
935   { thechapter } \l__postnotes_print_chapter_tl
936   \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
937   { thesection } \l__postnotes_print_section_tl
938   \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
939   { pnsectname } \l__postnotes_print_sectname_tl
940   \tl_set_eq:NN \pntheppage \l__postnotes_print_page_tl
941   \__postnotes_prop_get:nnN
942   { \l__postnotes_print_note_id_tl }
943   { mark } \l__postnotes_print_mark_tl
944   \__postnotes_prop_get:nnN
945   { \l__postnotes_print_note_id_tl }
946   { counter } \l__postnotes_print_counter_tl
947   \__postnotes_prop_get:nnN
948   { \l__postnotes_print_note_id_tl }
949   { content } \l__postnotes_print_content_tl
950   \tl_set:Nn \@currentcounter { postnotetext }
951   \int_set:Nn \c@postnotetext
952   { \l__postnotes_print_counter_tl }
953   \protected@edef \@currentlabel

```

```

954         { \p@postnote \l__postnotes_print_mark_tl }
955     \tl_set:Nn \l__postnotes_print_typeset_mark_tl
956     {
957         \tag_socket_use:n { postnotes/printmark/begin }
958         \UseHook { postnotes/print/note/typesetmark }
959         \MakeLinkTarget*
960         { postnote. \l__postnotes_print_note_id_tl .text }
961         \l__postnotes_pre_textmark_tl
962         \__postnotes_typeset_text_mark:eV
963         { \l__postnotes_print_note_id_tl }
964         \l__postnotes_print_mark_tl
965         \l__postnotes_post_textmark_tl
966         \tag_socket_use:n { postnotes/printmark/end }
967     }

```

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).

```

968     \bool_if:NTF \l__postnotes_print_as_list_bool
969     {
970         \item
971         [
972             \tag_socket_use:n { postnotes/printnote/begin }
973             \l__postnotes_print_typeset_mark_tl
974         ]
975         \mode_leave_vertical:
976     }
977     {
978         \mode_leave_vertical:
979         \tag_socket_use:n { postnotes/printnote/begin }
980         \l__postnotes_print_typeset_mark_tl
981     }

```

In principle, inserting the `ltmarks` marks would be best done at `\l__postnotes_print_typeset_mark_tl`, immediately *before* the mark, alongside the anchor. However, `\mark_insert:nn` is documented not to work inside a box (see <https://tex.stackexchange.com/a/732007>) and the `\item`'s label is typeset as a box, so that's not an option. Before the `\item` is also not a good idea. As long as the mark is no longer than a line, which is arguably to be expected, there shouldn't be any difference though. Either way, I don't see any other placement options.

```

982     \UseHook { postnotes/print/note/ltmarks }
983     \mark_insert:nn { postnotes/page }
984     { \l__postnotes_print_page_tl }
985     \mark_insert:nn { postnotes/chapter }
986     { \l__postnotes_print_chapter_tl }
987     \mark_insert:nn { postnotes/section }
988     { \l__postnotes_print_section_tl }
989     \mark_insert:nn { postnotes/sectname }
990     { \l__postnotes_print_sectname_tl }
991     \tag_socket_use:n { postnotes/printtext/begin }

```

```

992         \l__postnotes_print_content_tl
993         \tag_socket_use:n { postnotes/printtext/end }
994         \tag_socket_use:n { postnotes/printnote/end }
995         \l__postnotes_post_printnote_tl
996     \group_end:
997     \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
998     {
999         \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).

```

1000         \bool_if:NTF \l__postnotes_print_as_list_bool
1001         {
1002             \exp_args:NV \end \l__postnotes_print_env_tl
1003             \par
1004         }
1005         {
1006             \par
1007             \group_end:
1008         }
1009     }

```

Set `type_prev` for the next iteration.

```

1010         \tl_set:NV \l__postnotes_print_type_prev_tl
1011         \l__postnotes_print_type_curr_tl
1012     }
1013 }
1014 \__postnotes_set_header_vars_bool:

```

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

```

1015     \seq_map_inline:Nn \l__postnotes_clear_queue_seq
1016     { \__postnotes_prop_gclear:n { ##1 } }
1017 }
1018 \group_end:
1019 }

```

*(End of definition for `\__postnotes_print_notes:.`)*

```

1020 \msg_new:nnn { postnotes } { empty-printpostnotes }
1021 { 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>}
1022 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
1023 {
1024     \bool_lazy_and:nnTF
1025     { \l__postnotes_hyperlink_bool }
1026     { \l__postnotes_backlink_bool }
1027     {
1028         \__postnotes_make_text_mark:nnn {#2}
1029         { \hyper@linkstart { link } { postnote. #1 .mark } }
1030         { \hyper@linkend }
1031     }
1032     { \__postnotes_make_text_mark:nnn {#2} { } { } }

```

```

1033 }
1034 \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 `countersaux`, the numbering of floats are set at “shipout time”, so they belong where they are typeset. In other words, with `countersaux` 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 `countersaux` 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 `countersaux` the cut is made at the place the `print` 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:`

```

1035 \seq_new:N \g__postnotes_print_labelseq_queue_seq
1036 \cs_new_protected:Npn \__postnotes_split_labelseq:
1037 {
1038   \group_begin:
1039     \seq_clear:N \l__postnotes_tmpa_seq
1040     \seq_clear:N \l__postnotes_tmpb_seq
1041     \bool_if:NTF \g__postnotes_countersaux_bool
1042     {
1043       \tl_set:Nc \l__postnotes_tmpa_tl
1044         { { print } { \int_use:N \g__postnotes_print_postnotes_int } }

```

The `print` 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 prints in the sequence.

```

1045     \seq_if_in:NVF \g__postnotes_labelseq_seq \l__postnotes_tmpa_tl
1046     { \seq_gput_right:NV \g__postnotes_labelseq_seq \l__postnotes_tmpa_tl }
1047     \bool_do_until:nn
1048     { \tl_if_eq_p:NN \l__postnotes_tmpb_tl \l__postnotes_tmpa_tl }
1049     {
1050       \seq_gpop_left:NN \g__postnotes_labelseq_seq \l__postnotes_tmpb_tl
1051       \str_case:enT
1052       { \tl_item:Nn \l__postnotes_tmpb_tl { 1 } }
1053       {
1054         { mark } { }
1055         { section } { }
1056       }
1057     }

```

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.

```

1058         \int_compare:nNnTF
1059         { \tl_item:Nn \l__postnotes_tmpb_tl { 2 } }
1060         >
1061         { \g__postnotes_note_id_int }
1062         {
1063         \seq_put_right:Ne \l__postnotes_tmpb_seq
1064         \l__postnotes_tmpb_tl
1065         }
1066         {
1067         \seq_put_right:Ne \l__postnotes_tmpa_seq
1068         { \tl_item:Nn \l__postnotes_tmpb_tl { 2 } }
1069         }
1070     }

```

No need for the `F` branch of `\str_case:enT`, at this point the `print` label 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.

```

1071     }
1072     \seq_gset_eq:NN \g__postnotes_print_labelseq_queue_seq
1073     \l__postnotes_tmpa_seq
1074     \seq_concat:NNN \l__postnotes_tmpa_seq \l__postnotes_tmpb_seq
1075     \g__postnotes_labelseq_seq
1076     \seq_gset_eq:NN \g__postnotes_labelseq_seq \l__postnotes_tmpa_seq
1077 }
1078 {
1079     \seq_map_inline:Nn \g__postnotes_labelseq_seq
1080     {
1081         \str_case:enT
1082         { \tl_item:nn { ##1 } { 1 } }
1083         {
1084             { mark } { }
1085             { section } { }
1086         }
1087         {
1088             \int_compare:nNnTF
1089             { \tl_item:nn { ##1 } { 2 } }
1090             >
1091             { \g__postnotes_note_id_int }
1092             { \seq_put_right:Nn \l__postnotes_tmpb_seq { ##1 } }
1093             {
1094                 \seq_put_right:Ne \l__postnotes_tmpa_seq
1095                 { \tl_item:nn { ##1 } { 2 } }
1096             }
1097         }

```

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

```

1098     }

```

```

1099         \seq_gset_eq:NN \g__postnotes_print_labelseq_queue_seq
1100         \l__postnotes_tmpa_seq
1101         \seq_gset_eq:NN \g__postnotes_labelseq_seq \l__postnotes_tmpb_seq
1102     }
1103     \group_end:
1104 }

```

(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)
1105 \cs_new_protected:Npn \__postnotes_check_duplicates:N #1
1106 {
On a first run, don't even try to check for duplicates. Better transitorily let some
duplicates pass than to drop every legitimate note.
1107     \bool_lazy_and:nnT
1108     { ! \g__postnotes_firstrun_bool }
1109     { ! \g__postnotes_countersaux_bool }
1110     {
1111         \group_begin:
1112         \seq_clear:N \l__postnotes_tmpa_seq
1113         \seq_map_inline:Nn #1
1114         {
1115             \bool_lazy_or:nnTF
1116             { \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.

```

1117         {
1118             \str_if_eq_p:ee
1119             { \__postnotes_prop_item:nn {##1} { type } } { section }
1120         }
1121         { \seq_put_right:Nn \l__postnotes_tmpa_seq {##1} }
1122         {

```

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

```

1123         \str_if_eq:eeF
1124         { \__postnotes_prop_item:nn {##1} { multibool } }
1125         { true }

```

```

1126         {
1127             \seq_put_right:Nn \l__postnotes_tmpa_seq {##1}
1128             \bool_if:NT \l__postnotes_check_dupli_bool
1129             {
1130                 \msg_warning:nne { postnotes } { possible-duplicate }
1131                 { \l__postnotes_prop_item:nn {##1} { mark } }
1132             }
1133         }
1134     }
1135 }
1136 \seq_gset_eq:NN #1 \l__postnotes_tmpa_seq
1137 \group_end:
1138 }
1139 }

```

(End of definition for `\__postnotes_check_duplicates:N`.)

```

1140 \msg_new:nnn { postnotes } { possible-duplicate }
1141 { Possible~duplicate~*around*~note~'#1'~\msg_line_context:. }

```

`\__postnotes_check_floats:N`

```

\__postnotes_check_floats:N <\g__postnotes_print_queue_seq>
1142 \cs_new_protected:Npn \__postnotes_check_floats:N #1
1143 {
1144     \bool_lazy_and:nnT
1145     { \l__postnotes_check_floats_bool }

```

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

```

1146     { ! \g__postnotes_firstrun_bool }
1147     {
1148         \group_begin:

```

Only compare sequence net of non-existing labels.

```

1149         \seq_set_filter:NNn \l__postnotes_tmpa_seq #1
1150         {
1151             \bool_lazy_or_p:mn
1152             { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ mark @ ##1 } }
1153             { \cs_if_exist_p:c { \c__postnotes_ref_prefix_tl @ section @ ##1 } }
1154         }

```

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.

```

1155         \tl_if_eq:NNF
1156         \g__postnotes_print_labelseq_queue_seq \l__postnotes_tmpa_seq
1157         { \msg_warning:nn { postnotes } { possible-shuffle } }
1158     \group_end:
1159 }
1160 }

```

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

```

1161 \msg_new:nnn { postnotes } { possible-shuffle }
1162 { 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>

```

```

1163 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
1164 {
1165   \bool_if:NT \l__postnotes_sort_bool
1166   {
1167     \group_begin:
1168     \seq_gsort:Nn #1
1169     {
1170       \__postnotes_prop_get:nnN {##1} { psectid } \l__postnotes_tmpa_tl
1171       \__postnotes_prop_get:nnN {##2} { psectid } \l__postnotes_tmpb_tl
1172       \tl_if_eq:NNTF \l__postnotes_tmpa_tl \l__postnotes_tmpb_tl
1173       {
1174         \__postnotes_prop_get:nnN {##1} { type } \l__postnotes_tmpa_tl
1175         \__postnotes_prop_get:nnN {##2} { type } \l__postnotes_tmpb_tl
1176         \bool_lazy_and:nnTF
1177         { \str_if_eq_p:Vn \l__postnotes_tmpa_tl { note } }
1178         { \str_if_eq_p:Vn \l__postnotes_tmpb_tl { note } }
1179         {
1180           \__postnotes_prop_get:nnN {##1}
1181           { sortnum } \l__postnotes_tmpa_tl
1182           \__postnotes_prop_get:nnN {##2}
1183           { sortnum } \l__postnotes_tmpb_tl
1184           \fp_compare:nNnTF
1185           { \l__postnotes_tmpa_tl } > { \l__postnotes_tmpb_tl }
1186           { \sort_return_swapped: }
1187           { \sort_return_same: }
1188         }
1189         { \sort_return_same: }
1190       }
1191       { \sort_return_same: }
1192     }
1193   \group_end:
1194 }
1195 }

```

(End of definition for \\_\_postnotes\_sort\_queue:N.)

```

1196 \AddToHook { enddocument/afterlastpage }
1197 {
1198   \group_begin:
1199   \bool_if:NNTF \g__postnotes_countersaux_bool
1200   {
1201     \seq_set_filter:NNn \l__postnotes_tmpa_seq \g__postnotes_labelseq_seq
1202     { \str_if_eq_p:ee { \tl_item:nn {#1} { 1 } } { mark } }
1203   }
1204   {
1205     \seq_set_filter:NNn \l__postnotes_tmpa_seq \g__postnotes_queue_seq
1206     { \str_if_eq_p:ee { \__postnotes_prop_item:nn {#1} { type } } { note } }
1207   }
1208   \seq_if_empty:NF \l__postnotes_tmpa_seq
1209   {
1210     \msg_warning:nne { postnotes } { stray-notes }
1211     { \seq_count:N \l__postnotes_tmpa_seq }
1212   }
1213   \group_end:
1214 }

```

```

1215 \msg_new:nnn { postnotes } { stray-notes }
1216 {
1217   There-are-’#1’-stray-notes-after-the-last-’\iow_char:N\printpostnotes’-
1218   \msg_line_context:.-At-this-point,-they-are-lost.
1219 }

```

## 8 Headers

NOTE The use of these variables has been deprecated in 2024-12-03 for v0.5.0. The setting of values for them, now based on `ltmarks` data, is still transitionally provided so that users have time to adjust to the new mechanism without immediate breakage of existing documents. These variables are scheduled to be removed though, and users *must* migrate. Not only that, these values have been less intensively tested, and also require one compilation run more to converge. So there’s also a clear benefit in migrating sooner rather than later.

```

1220 \bool_new:N \l__postnotes_deprecated_headervars_bool
1221 \keys_define:nn { postnotes/setup }
1222 {
1223   deprecatedheadervars .bool_set:N = \l__postnotes_deprecated_headervars_bool ,
1224   deprecatedheadervars .default:n = true ,
1225   deprecatedheadervars .initial:n = false ,
1226 }
1227 \AddToHook { begindocument }
1228 {
1229   \keys_define:nn { postnotes/setup }
1230   {
1231     deprecatedheadervars .code:n =
1232     {
1233       \msg_warning:nnn { postnotes }
1234       { option-preamble-only } { deprecatedheadervars }
1235     } ,
1236   }
1237 }
1238 \tl_new:N \pnhdpagefirst
1239 \tl_new:N \pnhdpagelast
1240 \tl_new:N \pnhdchapfirst
1241 \tl_new:N \pnhdchaplast
1242 \tl_new:N \pnhdsectfirst
1243 \tl_new:N \pnhdsectlast
1244 \tl_new:N \pnhdnamefirst
1245 \tl_new:N \pnhdnamelast
1246 \tl_gset:Nn \pnhdpagefirst
1247 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdpagefirst } }
1248 \tl_gset:Nn \pnhdpagelast
1249 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdpagelast } }
1250 \tl_gset:Nn \pnhdchapfirst
1251 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdchapfirst } }
1252 \tl_gset:Nn \pnhdchaplast
1253 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdchaplast } }
1254 \tl_gset:Nn \pnhdsectfirst
1255 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdsectfirst } }
1256 \tl_gset:Nn \pnhdsectlast

```

```

1257 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdsectlast } }
1258 \tl_gset:Nn \pnhdnamefirst
1259 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdnamefirst } }
1260 \tl_gset:Nn \pnhdnamelast
1261 { \msg_error:nnn { postnotes } { deprecatedheadervars } { pnhdnamelast } }
1262 \msg_new:nnn { postnotes } { deprecatedheadervars }
1263 {
1264   '\iow_char:N\ #1'~is~deprecated~\msg_line_context:~
1265   You~should~migrate~to~the~new~'ltmarks'~(see~User~manual),~but~for~
1266   a~temporary~quick~fix~you~can~enable~the~'deprecatedheadervars'~option.
1267 }
1268 \cs_set_eq:NN \__postnotes_set_header_vars_first: \prg_do_nothing:
1269 \cs_set_eq:NN \__postnotes_set_header_vars_bool: \prg_do_nothing:
1270 \AddToHook { begindocument }
1271 {
1272   \bool_if:NT \l__postnotes_deprecated_headervars_bool
1273   {
1274     \property_new:nnnn { postnotes/page/first } { shipout } { }
1275     { \FirstMark { postnotes/page } }
1276     \property_new:nnnn { postnotes/page/last } { shipout } { }
1277     { \LastMark { postnotes/page } }
1278     \property_new:nnnn { postnotes/chapter/first } { shipout } { }
1279     { \FirstMark { postnotes/chapter } }
1280     \property_new:nnnn { postnotes/chapter/last } { shipout } { }
1281     { \LastMark { postnotes/chapter } }
1282     \property_new:nnnn { postnotes/section/first } { shipout } { }
1283     { \FirstMark { postnotes/section } }
1284     \property_new:nnnn { postnotes/section/last } { shipout } { }
1285     { \LastMark { postnotes/section } }
1286     \property_new:nnnn { postnotes/sectname/first } { shipout } { }
1287     { \FirstMark { postnotes/sectname } }
1288     \property_new:nnnn { postnotes/sectname/last } { shipout } { }
1289     { \LastMark { postnotes/sectname } }
1290     \clist_const:Nn \c__postnotes_header_marks_clist
1291     {
1292       postnotes/page/first ,
1293       postnotes/page/last ,
1294       postnotes/chapter/first ,
1295       postnotes/chapter/last ,
1296       postnotes/section/first ,
1297       postnotes/section/last ,
1298       postnotes/sectname/first ,
1299       postnotes/sectname/last ,
1300     }
1301     \cs_new_protected:Npn \__postnotes_set_header_vars:n #1
1302     {
1303       \group_begin:
1304         \protected@xdef \pnhdpagefirst
1305         { \property_ref:nn { #1 } { postnotes/page/first } }
1306         \protected@xdef \pnhdpagelast
1307         { \property_ref:nn { #1 } { postnotes/page/last } }
1308         \protected@xdef \pnhdchapfirst
1309         { \property_ref:nn { #1 } { postnotes/chapter/first } }
1310         \protected@xdef \pnhdchaplast

```

```

1311         { \property_ref:nn { #1 } { postnotes/chapter/last } }
1312     \protected@xdef \pnhdsectfirst
1313         { \property_ref:nn { #1 } { postnotes/section/first } }
1314     \protected@xdef \pnhdsectlast
1315         { \property_ref:nn { #1 } { postnotes/section/last } }
1316     \protected@xdef \pnhdnamefirst
1317         { \property_ref:nn { #1 } { postnotes/sectname/first } }
1318     \protected@xdef \pnhdnamelast
1319         { \property_ref:nn { #1 } { postnotes/sectname/last } }
1320     \group_end:
1321 }
1322 \cs_generate_variant:Nn \__postnotes_set_header_vars:n { e }
1323 \bool_new:N \g__postnotes_header_vars_next_bool
1324 \int_new:N \g__postnotes_print_header_vars_int
1325 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:
1326 {
1327     \bool_if:NT \g__postnotes_header_vars_next_bool
1328     {
1329         \int_gincr:N \g__postnotes_print_header_vars_int
1330         \exp_args:Ne \property_record:nN
1331             {
1332                 __postnotes_header_
1333                 \int_use:N \g__postnotes_print_header_vars_int
1334             }
1335         \c__postnotes_header_marks_clist
1336         \__postnotes_set_header_vars:e
1337         {
1338             __postnotes_header_
1339             \int_use:N \g__postnotes_print_header_vars_int
1340         }
1341     }
1342 }
1343 \AddToHook { shipout/before } [ ./header ]
1344 { \__postnotes_set_headers_vars_next: }
1345 \cs_set_protected:Npn \__postnotes_set_header_vars_first:
1346 {
1347     \bool_gset_true:N \g__postnotes_header_vars_next_bool
1348     \tl_gset:Nn \pnhdpagefirst { \FirstMark{postnotes/page} }
1349     \tl_gset:Nn \pnhdpagelast { \LastMark{postnotes/page} }
1350     \tl_gset:Nn \pnhdchapfirst { \FirstMark{postnotes/chapter} }
1351     \tl_gset:Nn \pnhdchaplast { \LastMark{postnotes/chapter} }
1352     \tl_gset:Nn \pnhdsectfirst { \FirstMark{postnotes/section} }
1353     \tl_gset:Nn \pnhdsectlast { \LastMark{postnotes/section} }
1354     \tl_gset:Nn \pnhdnamefirst { \FirstMark{postnotes/sectname} }
1355     \tl_gset:Nn \pnhdnamelast { \LastMark{postnotes/sectname} }
1356     \int_gincr:N \g__postnotes_print_header_vars_int
1357     \exp_args:Ne \property_record:nN
1358         {
1359             __postnotes_header_
1360             \int_use:N \g__postnotes_print_header_vars_int
1361         }
1362     \c__postnotes_header_marks_clist
1363     \__postnotes_set_header_vars:e
1364     {

```

```

1365         __postnotes_header_
1366         \int_use:N \g__postnotes_print_header_vars_int
1367     }
1368 }
1369 \cs_set_protected:Npn \__postnotes_set_header_vars_bool:
1370 {
1371     \AddToHookNext { shipout/after }
1372     { \bool_gset_false:N \g__postnotes_header_vars_next_bool }
1373 }
1374 }
1375 }

```

## 9 Compatibility

A dedicated temp variable for restoring data.

```

1376 \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.

```

1377 \AddToHook { postnotes/note/begin } [ ./compat/caption ]
1378 { \cs_if_exist:NT \@captype { \postnotesetup { maybemulti } } }

```

### **biblatex**

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

```

1379 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1380 {

```

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

```

1381     \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1382     { \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.

```

1383     \AddToHook { begindocument/before } [ postnotes/compat/biblatex ]
1384     {
1385         \cs_set:Npn \blx@theendnote { \postnote }
1386         \cs_set:Npn \blx@theendnotetext { \blx@err@endnote \footnotetext }
1387     }
1388 }

```



1389 `<*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):

```
1390 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1391 {
```

Store biblatex variables for each note.

```
1392   \AddToHook { postnotes/note/store } [ postnotes/compat/biblatex ]
1393   {
1394     \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1395     { biblatex@refsection } { \int_use:N \c@refsection }
1396     \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1397     { biblatex@refsegment } { \int_use:N \c@refsegment }
1398     \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1399     { biblatex@refcontextbool }
1400     { \iftoggle { blx@refcontext } { true } { false } }
1401     \prop_gput:cnV { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1402     { biblatex@refcontext } \blx@refcontext@context
1403   }
```

biblatex setup, once for `\printpostnotes` call.

```
1404   \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1405   {
1406     \__postnotes_biblatex_endrefcontext_local:
1407     \__postnotes_biblatex_citereset_local:
1408   }
```

Restore biblatex variables for each note.

```
1409   \AddToHook { postnotes/print/note/begin } [ postnotes/compat/biblatex ]
1410   {
1411     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1412     { biblatex@refsection } \l__postnotes_restore_tmp_tl
1413     \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
1414     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1415     { biblatex@refsegment } \l__postnotes_restore_tmp_tl
1416     \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
1417     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1418     { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
1419     \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
1420     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1421     { biblatex@refcontext } \l__postnotes_restore_tmp_tl
1422     \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
1423   }
```

Auxiliary functions.

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

```
1424 \cs_new_protected:Npn \_postnotes_biblatex_endrefcontext_local:
1425 {
1426   \togglefalse { blx@refcontext }
1427   \tl_clear:N \blx@refcontext@labelprefix
1428   \tl_clear:N \blx@refcontext@labelprefix@real
1429   \tl_set:Ne \blx@refcontext@sortingtemplatename { \blx@sorting }
1430   \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
1431   \tl_set:Nn \blx@refcontext@uniquenametemplatenamename { global }
1432   \tl_set:Nn \blx@refcontext@labelalphanametemplatenamename { global }
1433   \blx@edef@refcontext
1434   {
1435     \blx@refcontext@sortingtemplatename /
1436     \blx@refcontext@sortingnamekeytemplatename /
1437     /
1438     \blx@refcontext@uniquenametemplatenamename /
1439     \blx@refcontext@labelalphanametemplatenamename
1440   }
1441 }
```

(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`.

```
1442 \cs_new_protected:Npn \_postnotes_biblatex_citereset_local:
1443 {
1444   \global\cslet{blx@bsee@\the\c@refsection}\@empty
1445   \global\cslet{blx@fsee@\the\c@refsection}\@empty
1446   \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1447   \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
1448   \blx@ibidreset@force
1449   \undef \blx@lastkey@text
1450   \undef \blx@lastkey@foot
1451   \blx@idemreset@force
1452   \undef \blx@lasthash@text
1453   \undef \blx@lasthash@foot
1454   \blx@opcitreset@force
1455   \clist_map_inline:Nn \blx@trackhash@text
1456   { \csundef { blx@lastkey@text@ ##1 } }
1457   \tl_clear:N \blx@trackhash@text
1458   \clist_map_inline:Nn \blx@trackhash@foot
1459   { \csundef { blx@lastkey@foot@ ##1 } }
1460   \tl_clear:N \blx@trackhash@foot
1461   \blx@loccitreset@force
1462   \clist_map_inline:Nn \blx@trackkeys@text
1463   { \csundef { blx@lastnote@text@ ##1 } }
1464   \tl_clear:N \blx@trackkeys@text
```

```

1459     \clist_map_inline:Nn \blx@trackkeys@foot
1460     { \csundef { blx@lastnote@foot@ ##1 } }
1461     \tl_clear:N \blx@trackkeys@foot

```

and all of them do:

```

1462     \cs_set_eq:NN \blx@lastmpfn \z@
1463     }

```

(End of definition for \\_\_postnotes\_biblatex\_citereset\_local:.)

```

1464 }

```

biblatex’s `refsections`, contrary to `refsegments` and `refcontexts` which are handled in the  $\LaTeX$  side of things (as far as I can tell), need to go through `biber`, and must have correct corresponding citation data written to the `.bcf` file. And the way `\refsection` is implemented presumes each section is only ever begun once (fair...), thus making it difficult to “reopen” it, or append new citations to it later on, when the notes are printed. The start of a `refsection` must be registered on the `.bcf` file, and this is done by `\refsection` (and its auxiliary functions). However, a number of its characteristics make things particularly difficult for the purpose at hand: i) it unconditionally sets a label for the section which, of course, cannot be done twice; and, critically, ii) the optional argument of the environment (which receives the  $\langle resources \rangle$ ) 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`.

```

1465 \AddToHook { package/biblatex/after } [ ./compat/biblatex ]
1466 {
1467     \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1468     \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1469     \AddToHook { postnotes/print/begin } [ postnotes/compat/biblatex ]
1470     {
1471         \tl_set:Ne \l__postnotes_biblatex_orig_refsection_tl
1472         { \int_use:N \c@refsection }
1473         \tl_gset:Ne \g__postnotes_biblatex_prev_refsection_tl
1474         { \l__postnotes_biblatex_orig_refsection_tl }
1475     }
1476     \AddToHook { postnotes/print/note/begin } [ postnotes/compat/biblatex ]
1477     {
1478         \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1479         { biblatex@refsection } \l__postnotes_restore_tmp_tl
1480         \tl_if_eq:NNF
1481         \l__postnotes_restore_tmp_tl
1482         \g__postnotes_biblatex_prev_refsection_tl
1483         {
1484             \int_set:Nn \c@blx@maxsection
1485             { \l__postnotes_restore_tmp_tl - 1 }
1486             \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1487             \l__postnotes_restore_tmp_tl
1488             \group_begin:
1489             \cs_set_eq:NN \label \use_none:n
1490             \cs_set_eq:NN \blx@info \use_none:n

```

```

1491         \blx@endrefsection
1492         \refsection
1493         \group_end:
1494     }
1495 }
1496 }
1497 </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.

```

1498 \str_new:N \l__postnotes_note_zlabel_str

```

*(End of definition for \l\_\_postnotes\_note\_zlabel\_str.)*

```

1499 \AddToHook { package/zref-user/after } [ ./compat/zref-user ]
1500 {

```

Provide `zlabel` option.

```

1501     \keys_define:nn { postnotes/note }
1502     {
1503         zlabel .str_set:N = \l__postnotes_note_zlabel_str ,
1504         zlabel .value_required:n = true ,
1505     }

```

Provide property to store the mark for measuring passes.

```

1506     \zref@newprop { postnote@mark } [] { \l__postnotes_mark_tl }
1507     \AddToHook { postnotes/note/begin } [ postnotes/compat/zref-user ]
1508     { \zref@localaddprop { main } { postnote@mark } }

```

`\postnotezref` Provide `\postnotezref`.

```

\postnotezref(*)<{label}>

```

```

1509 \NewDocumentCommand \postnotezref { s m }
1510 { \__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}>

```

```

1511 \str_new:N \l__postnotes_note_zref_zlabel_str
1512 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1513 {
1514     \group_begin:
1515     \str_set:Nn \l__postnotes_note_zref_zlabel_str {#2}
1516     \__postnotes_typeset_mark_wrapper:nnn
1517     {
1518         \bool_lazy_all:nTF
1519         {
1520             { ! #1 }

```

```

1521         { \l__postnotes_hyperlink_bool }
1522         { \l__postnotes_zrefhyperref_bool }
1523     }
1524     {
1525     \hyperlink
1526     { \zref@extractdefault {#2} { anchor } { } }
1527     { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1528     }
1529     { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1530     }
1531     { \tag_socket_use:n { postnotes/postnotetzref/begin } }
1532     { \tag_socket_use:n { postnotes/postnotetzref/end } }
1533 \group_end:
1534 }

```

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

```

1535 }
1536 \bool_new:N \l__postnotes_zrefhyperref_bool
1537 \AddToHook { package/zref-hyperref/after } [ ./compat/zref-hyperref ]
1538 { \bool_set_true:N \l__postnotes_zrefhyperref_bool }

```

## zref-clever

```

1539 \AddToHook { package/zref-clever/after } [ ./compat/zref-clever ]
1540 {
1541     \zcsetup
1542     {
1543         countertype = { postnote = endnote } ,
1544         countertype = { postnotetext = endnote } ,
1545     }
1546     \AddToHook { postnotes/print/begin } [ postnotes/compat/zref-clever ]
1547     { \zcsetup { counterresetby = { postnotetext = postnotesection } } }
1548 }

```

## zref-check

```

1549 \AddToHook { package/zref-check/after } [ ./compat/zref-check ]
1550 {
1551     \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1552     {
1553         \AddToHook { postnotes/note/store } [ postnotes/compat/zref-check ]
1554         {
1555             \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1556             { zref-check@abschap } { \int_use:N \c@zc@abschap }
1557             \prop_gput:cne { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1558             { zref-check@abssec } { \int_use:N \c@zc@abssec }
1559         }
1560         \AddToHook { postnotes/print/note/begin } [ postnotes/compat/zref-check ]
1561         {
1562             \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1563             { zref-check@abschap } \l__postnotes_restore_tmp_tl
1564             \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1565             \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1566             { zref-check@abssec } \l__postnotes_restore_tmp_tl

```

```

1567         \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1568     }
1569 }
1570 { }
1571 }

```

## amsmath

```

1572 \AddToHook { package/amsmath/after } [ ./compat/amsmath ]
1573 {

```

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

```

1574     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/amsmath ]
1575     {
1576         \legacy_if:nT { measuring@ }
1577         {
1578             \bool_set_true:N \l__postnotes_inhibit_note_bool
1579             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1580             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1581         }
1582     }

```

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.)”

```

1583     \AddToHook { postnotes/note/begin } [ postnotes/compat/amsmath ]
1584     { \legacy_if:nF { firstchoice@ } { \postnotesetup { maybemulti } } }
1585 }

```

## csquotes

```

1586 \AddToHook { package/csquotes/after } [ ./compat/csquotes ]
1587 {
1588     \bool_new:N \l__postnotes_csquotes_measuring_bool
1589     \BlockquoteDisable
1590     { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1591     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/csquotes ]
1592     {
1593         \bool_if:NT \l__postnotes_csquotes_measuring_bool
1594         {
1595             \bool_set_true:N \l__postnotes_inhibit_note_bool
1596             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1597             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1598         }
1599     }
1600 }

```

## 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>.

```
1601 \AddToHook { package/tabularx/after } [ ./compat/tabularx ]
1602   {
1603     \bool_new:N \l__postnotes_tabularx_inside_env_bool
1604     \AddToHook { env/tabularx/begin } [ postnotes/compat/tabularx ]
1605     {
1606       \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1607       \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1608     }
1609     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/tabularx ]
1610     {
1611       \bool_lazy_and:nnT
1612         { \l__postnotes_tabularx_inside_env_bool }
1613         { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1614         {
1615           \bool_set_true:N \l__postnotes_inhibit_note_bool
1616           \bool_set_true:N \l__postnotes_print_plain_mark_bool
1617           \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1618         }
1619     }
1620     \AddToHook { package/xltabular/after } [ postnotes/compat/xltabular ]
1621     {
1622       \AddToHook { env/xltabular/begin } [ postnotes/compat/xltabular ]
1623       {
1624         \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1625         \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1626       }
1627     }
1628   }
```

## tabularray

```
1629 \AddToHook { package/tabularray/after } [ ./compat/tabularray ]
1630   {
```

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).

```
1631     \bool_if_exist:NT \lTblrMeasuringBool
1632     {
```

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.

```

1633     \AddToHook { postnotes/note/inhibit } [ postnotes/compat/tabularray ]
1634     {
1635         \bool_if:NT \lTblrMeasuringBool
1636         {
1637             \bool_set_true:N \l__postnotes_inhibit_note_bool
1638             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1639             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1640         }
1641     }
1642 }
1643 }

```

## 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 `\label` 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 `\label` 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 `\label` 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 `\label` argument, as we did for the case of labels inside the note. Auxiliary labels are handled with `lproperties` labels.

Unconditionally define tagging support sockets.

```

1644 \socket_new:nn { tagssupport/postnotes/mark/begin }{ 0 }
1645 \socket_new:nn { tagssupport/postnotes/mark/end }{ 0 }
1646 \socket_new:nn { tagssupport/postnotes/nomark/begin }{ 0 }
1647 \socket_new:nn { tagssupport/postnotes/nomark/end }{ 0 }
1648 \socket_new:nn { tagssupport/postnotes/multisep/begin }{ 0 }
1649 \socket_new:nn { tagssupport/postnotes/multisep/end }{ 0 }
1650 \socket_new:nn { tagssupport/postnotes/printlist/begin }{ 0 }
1651 \socket_new:nn { tagssupport/postnotes/printlist/end }{ 0 }
1652 \socket_new:nn { tagssupport/postnotes/printnote/begin }{ 0 }
1653 \socket_new:nn { tagssupport/postnotes/printnote/end }{ 0 }
1654 \socket_new:nn { tagssupport/postnotes/printmark/begin }{ 0 }
1655 \socket_new:nn { tagssupport/postnotes/printmark/end }{ 0 }
1656 \socket_new:nn { tagssupport/postnotes/printtext/begin }{ 0 }
1657 \socket_new:nn { tagssupport/postnotes/printtext/end }{ 0 }
1658 \socket_new:nn { tagssupport/postnotes/postnoteref/begin }{ 0 }
1659 \socket_new:nn { tagssupport/postnotes/postnoteref/end }{ 0 }
1660 \socket_new:nn { tagssupport/postnotes/postnotezref/begin }{ 0 }
1661 \socket_new:nn { tagssupport/postnotes/postnotezref/end }{ 0 }

1662 \bool_lazy_and:nnT
1663   { \cs_if_exist_p:N \tag_if_active_p: }
1664   { \tag_if_active_p: }
1665   {

```

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

```

1666   \tagpdfsetup
1667   {
1668     role/new-tag = { tag=endnote, role=FENote } ,
1669     role/new-tag = { tag=endnotemark, role=Lbl } ,
1670     role/new-tag = { tag=endnotelabel, role=Lbl } ,
1671     role/new-attribute =
1672       { EndnoteType } { /0 /FENote /NoteType /Endnote } ,
1673     role/new-attribute =
1674       { EndnotesList } { /0 /List /ListNumbering /Ordered } ,
1675   }
\postnote
1676   \socket_new_plug:nnn { tagssupport/postnotes/mark/begin } { default }
1677   {

```

```

1678     \tag_mc_end_push:
1679     \tag_struct_begin:n
1680     {
1681         tag = endnotemark ,
1682         label = { postnotemark. \l_postnotes_note_id_tl } ,
1683         ref = { postnote. \l_postnotes_note_id_tl } ,
1684     }
1685     \__postnotes_tagsup_store_sstructnum:nN
1686     { postnotemark } \l_postnotes_note_id_tl
1687     \tag_mc_begin:n { }
1688 }
1689 \socket_new_plug:nnn { tagsupport/postnotes/mark/end } { default }
1690 {
1691     \tag_mc_end:
1692     \tag_struct_end: % endnotemark
1693     \tag_mc_begin_pop:n { }
1694 }
1695 \socket_new_plug:nnn { tagsupport/postnotes/nomark/begin } { default }
1696 {
1697     \tag_struct_begin:n
1698     {
1699         tag = NonStruct ,
1700         label = { postnotemark. \l_postnotes_note_id_tl } ,
1701         ref = { postnote. \l_postnotes_note_id_tl } ,
1702     }
1703     \__postnotes_tagsup_store_sstructnum:nN
1704     { postnotemark } \l_postnotes_note_id_tl
1705 }
1706 \socket_new_plug:nnn { tagsupport/postnotes/nomark/end } { default }
1707 { \tag_struct_end: } % NonStruct
1708 \socket_assign_plug:nn { tagsupport/postnotes/mark/begin } { default }
1709 \socket_assign_plug:nn { tagsupport/postnotes/mark/end } { default }
1710 \socket_assign_plug:nn { tagsupport/postnotes/nomark/begin } { default }
1711 \socket_assign_plug:nn { tagsupport/postnotes/nomark/end } { default }
multiple
1712 \socket_new_plug:nnn { tagsupport/postnotes/multisep/begin } { default }
1713 {
1714     \tag_mc_end_push:
1715     \tag_mc_begin:n { artifact }
1716 }
1717 \socket_new_plug:nnn { tagsupport/postnotes/multisep/end } { default }
1718 {
1719     \tag_mc_end:
1720     \tag_mc_begin_pop:n { }
1721 }
1722 \socket_assign_plug:nn { tagsupport/postnotes/multisep/begin } { default }
1723 \socket_assign_plug:nn { tagsupport/postnotes/multisep/end } { default }
\printpostnotes
1724 \socket_new_plug:nnn { tagsupport/postnotes/printlist/begin } { default }
1725 { \tag_tool:n { para/tagging=false } }
1726 \socket_new_plug:nnn { tagsupport/postnotes/printlist/end } { default }
1727 { }
1728 \socket_assign_plug:nn { tagsupport/postnotes/printlist/begin } { default }

```

```

1729 \socket_assign_plug:mn { tagssupport/postnotes/printlist/end } { default }
1730 \socket_new_plug:nnn { tagssupport/postnotes/printnote/begin } { default }
1731 {
1732   \bool_if:NF \l__postnotes_print_as_list_bool
1733   {
1734     \tag_struct_begin:n
1735     {
1736       tag = endnote ,
1737       attribute-class = EndnoteType ,
1738       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.
1739       ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1740     }
1741     \__postnotes_tagssup_store_sstructnum:nN
1742     { postnote } \l_postnotes_print_note_id_tl
1743   }
1744 }
1745 \socket_new_plug:nnn { tagssupport/postnotes/printnote/end } { default }
1746 {
1747   \bool_if:NF \l__postnotes_print_as_list_bool
1748   { \tag_struct_end: } % endnote
1749 }
1750 \socket_assign_plug:mn { tagssupport/postnotes/printnote/begin } { default }
1751 \socket_assign_plug:mn { tagssupport/postnotes/printnote/end } { default }
1752 \socket_new_plug:nnn { tagssupport/postnotes/printmark/begin } { default }
1753 {
1754   \bool_if:NF \l__postnotes_print_as_list_bool
1755   {
1756     \tag_struct_begin:n { tag=endnotelabel }
1757     \tag_mc_begin:n { tag=Lbl }
1758   }
1759 }
1760 \socket_new_plug:nnn { tagssupport/postnotes/printmark/end } { default }
1761 {
1762   \bool_if:NF \l__postnotes_print_as_list_bool
1763   {
1764     \tag_mc_end:
1765     \tag_struct_end: % endnotelabel
1766   }
1767 }
1768 \socket_assign_plug:mn { tagssupport/postnotes/printmark/begin } { default }
1769 \socket_assign_plug:mn { tagssupport/postnotes/printmark/end } { default }
1770 \socket_new_plug:nnn { tagssupport/postnotes/printtext/begin } { default }
1771 {
1772   \bool_if:NTF \l__postnotes_print_as_list_bool
1773   {
1774     \tag_struct_begin:n
1775     {
1776       tag = endnote ,
1777       attribute-class = EndnoteType ,
1778       label = { postnote. \l_postnotes_print_note_id_tl } ,

```

CHECK Ditto.

```

1779         ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1780     }
1781     \__postnotes_tagsup_store_sctructnum:nN
1782     { postnote } \l_postnotes_print_note_id_tl
1783     \tag_struct_begin:n { tag=text-unit }
1784     \tag_struct_begin:n { tag=text }
1785     \tag_tool:n { para/tagging=true }
1786     \tag_mc_begin:n { }
1787     }
1788     {
1789     \tag_struct_begin:n { tag=text-unit }
1790     \tag_struct_begin:n { tag=text }
1791     \tag_tool:n { para/tagging=true }
1792     \tag_mc_begin:n { }
1793     }
1794     }
1795     \socket_new_plug:nnn { tagsupport/postnotes/printtext/end } { default }
1796     {
1797     \bool_if:NTF \l__postnotes_print_as_list_bool
1798     {
1799     \tag_mc_end:
1800     \tag_tool:n { para/tagging=false }
1801     \tag_struct_end: % text
1802     \tag_struct_end: % text-unit
1803     \tag_struct_end: % endnote
1804     }
1805     {
1806     \tag_mc_end:
1807     \tag_tool:n { para/tagging=false }
1808     \tag_struct_end: % text
1809     \tag_struct_end: % text-unit
1810     }
1811     }
1812     \socket_assign_plug:nn { tagsupport/postnotes/printtext/begin } { default }
1813     \socket_assign_plug:nn { 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.

```

1814     \IfInstanceExistsT { blockenv } { list }
1815     {
1816     \DeclareInstance { blockenv } { postnoteslist } { display }
1817     {
1818     env-name      = postnoteslist ,
1819     tag-name      = L ,
1820     tag-class     = EndnotesList ,
1821     tagging-recipe = list ,
1822     inner-level-counter = ,
1823     level-increase = true ,
1824     setup-code    = ,
1825     block-instance = list ,
1826     inner-instance = postnoteslist ,
1827     }
1828     \DeclareInstanceCopy { blockenv }

```

```

1829     { postnoteslisthang } { postnoteslist }
1830 \EditInstance { blockenv } { postnoteslisthang }
1831     { env-name = postnoteslisthang }
1832 \DeclareInstance { list } { postnoteslist } { std }
1833     { item-instance = postnoteslist }
1834 \DeclareInstance { item } { postnoteslist } { std }
1835     {
1836     label-format = { \hspace { \labelsep } \normalfont ~ #1 } ,
1837     label-align = left ,
1838     }
1839 \RenewDocumentEnvironment { postnoteslist } { }
1840     {
1841     \UseInstance { blockenv } { postnoteslist }
1842     {
1843     leftmargin      = Opt ,
1844     label-width     = Opt ,
1845     item-indent     = .5\parindent ,
1846     rightmargin    = Opt ,
1847     parindent      = \parindent ,
1848     par-skip       = \parskip ,
1849     item-skip      = Opt ,
1850     beginsep       = .5\topsep ,
1851     begin-par-skip = .5\partopsep ,
1852     }
1853     }
1854     { \endblockenv }
1855 \RenewDocumentEnvironment { postnoteslisthang } { }
1856     {
1857     \UseInstance { blockenv } { postnoteslisthang }
1858     {
1859     leftmargin      = 1em ,
1860     label-width     = -\leftmargin ,
1861     item-indent     = -2\leftmargin ,
1862     rightmargin    = Opt ,
1863     parindent      = \parindent ,
1864     par-skip       = \parskip ,
1865     item-skip      = Opt ,
1866     beginsep       = .5\topsep ,
1867     begin-par-skip = .5\partopsep ,
1868     }
1869     }
1870     { \endblockenv }
1871     }

```

Setup for \label and \zlabel inside the note.

```

1872 \bool_new:N \l__postnotes_inside_note_bool
1873 \AddToHookWithArguments { label } [ postnotes/tagsup ]
1874     {
1875     \bool_if:NT \l__postnotes_inside_note_bool
1876     {
1877     \property_record:nn { postnote@label@innote. #1 }
1878     { postnotes/tagsup@noteid }
1879     }
1880     }
1881 \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]

```

```

1882     { \bool_set_true:N \l__postnotes_inside_note_bool }
1883 \property_new:nmmn { postnotes/tagstup@noteid } { now } { 0 }
1884     { \l_postnotes_print_note_id_tl }
1885 \AddToHook { package/zref-user/after } [ postnotes/tagstup ]
1886     {
1887     \zref@newprop { postnotes@tagstup@noteid } [ 0 ]
1888     { \l_postnotes_print_note_id_tl }
1889     \AddToHook { postnotes/print/note/begin } [ postnotes/tagstup ]
1890     { \zref@localaddprop { main } { postnotes@tagstup@noteid } }
1891     }

```

Setup for label and zlabel options.

```

1892 \AddToHook { postnotes/note/store } [ postnotes/tagstup ]
1893     {
1894     \str_if_empty:NF \l__postnotes_note_label_str
1895     {
1896     \prop_gput:cnV
1897     { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1898     { label } \l__postnotes_note_label_str
1899     }
1900     }
1901 \AddToHook { package/zref-user/after } [ postnotes/tagstup ]
1902     {
1903     \AddToHook { postnotes/note/store } [ postnotes/tagstup ]
1904     {
1905     \str_if_empty:NF \l__postnotes_note_zlabel_str
1906     {
1907     \prop_gput:cnV
1908     { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1909     { zlabel } \l__postnotes_note_zlabel_str
1910     }
1911     }
1912     }
1913 \AddToHook { postnotes/print/note/begin } [ postnotes/tagstup ]
1914     {
1915     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1916     { label } \l__postnotes_restore_tmp_tl
1917     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1918     {
1919     \exp_args:Ne \property_record:nn
1920     { postnote@label@option. \l__postnotes_restore_tmp_tl }
1921     { postnotes/tagstup@noteid }
1922     }
1923     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1924     { zlabel } \l__postnotes_restore_tmp_tl
1925     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1926     {
1927     \exp_args:Ne \property_record:nn
1928     { postnote@zlabel@option. \l__postnotes_restore_tmp_tl }
1929     { postnotes/tagstup@noteid }
1930     }
1931     }

```

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_sstructnum:nN`  
`\_postnotes_tagsup_store_crossref:nN`

```

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

```

```

1932   \prop_new:N \g__postnotes_tagsup_structnums_prop
1933   \cs_new_protected:Npn \_postnotes_tagsup_store_sstructnum:nN #1#2
1934   {
1935     \prop_gput:Nee \g__postnotes_tagsup_structnums_prop
1936     { #1 . #2 } { \tag_get:n { struct_num } }
1937   }
1938   \prop_new:N \g__postnotes_tagsup_crossrefs_prop
1939   \cs_new_protected:Npn \_postnotes_tagsup_store_crossref:nN #1#2
1940   {
1941     \prop_gput:Nee \g__postnotes_tagsup_crossrefs_prop
1942     { \tag_get:n { struct_num } } { #1 . #2 }
1943   }

```

(End of definition for `\_postnotes_tagsup_store_sstructnum:nN` and `\_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`.

```

1944   \cs_new_protected:Npn \_postnotes_tagsup_gput_ref:nn #1#2
1945   {
1946     \tag_if_active:T
1947     { \tag_struct_gput:ene {#1} {ref} { \tag_struct_object_ref:e {#2} } }
1948   }

```

(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.

```

1949   \AddToHook { tagpdf/finish/before } [ postnotes/tagsup ]
1950   {
1951     \prop_map_inline:Nn \g__postnotes_tagsup_crossrefs_prop
1952     {
1953       \_postnotes_tagsup_gput_ref:nn
1954       { \prop_item:Nn \g__postnotes_tagsup_structnums_prop {#2} }
1955       {#1}
1956     }
1957   }

```

`\postnoteref`

```

1958   \socket_new_plug:nnn { tagsupport/postnotes/postnoteref/begin } { default }
1959   {
1960     \tag_mc_end_push:
1961     \property_if_recorded:eeTF
1962     { postnote@label@innote. \l__postnotes_note_ref_label_str }

```



```

1963         { postnotes/tag-sup@noteid }
1964     {

```

Label coming from a \label inside the note.

```

1965         \tl_set:Nc \l__postnotes_tmpa_tl
1966         {
1967             \property_ref:ee
1968             { postnote@label@innote. \l__postnotes_note_ref_label_str }
1969             { postnotes/tag-sup@noteid }
1970         }
1971     \tag_struct_begin:n
1972     {
1973         tag = endnotemark ,
1974         ref = { postnote. \l__postnotes_tmpa_tl } ,
1975     }
1976     \__postnotes_tag-sup_store_crossref:nN
1977     { postnote } \l__postnotes_tmpa_tl
1978 }
1979 {
1980     \property_if_recorded:eeTF
1981     { postnote@label@option. \l__postnotes_note_ref_label_str }
1982     { postnotes/tag-sup@noteid }
1983     {

```

Label coming from a label option.

```

1984         \tl_set:Nc \l__postnotes_tmpa_tl
1985         {
1986             \property_ref:ee
1987             { postnote@label@option. \l__postnotes_note_ref_label_str }
1988             { postnotes/tag-sup@noteid }
1989         }
1990     \tag_struct_begin:n
1991     {
1992         tag = endnotemark ,
1993         ref = { postnotemark. \l__postnotes_tmpa_tl } ,
1994     }
1995     \__postnotes_tag-sup_store_crossref:nN
1996     { postnotemark } \l__postnotes_tmpa_tl
1997 }
1998 { \tag_struct_begin:n { tag = endnotemark } }
1999 }
2000 \tag_mc_begin:n { }
2001 }
2002 \socket_new_plug:nnn { tag-support/postnotes/postnoteref/end } { default }
2003 {
2004     \tag_mc_end:
2005     \tag_struct_end: % endnotemark
2006     \tag_mc_begin_pop:n { }
2007 }
2008 \socket_assign_plug:mn { tag-support/postnotes/postnoteref/begin } { default }
2009 \socket_assign_plug:mn { tag-support/postnotes/postnoteref/end } { default }

```

\postnotezref

```

2010 \AddToHook { package/zref-user/after } [ postnotes/tag-sup ]
2011 {

```

```

2012     \socket_new_plug:nnn { tagsupport/postnotes/postnotezref/begin } { default }
2013     {
2014         \tag_mc_end_push:
2015         \zref@ifrefcontainsprop { \l__postnotes_note_zref_zlabel_str }
2016         { postnotes@tagsup@noteid }
2017         {

```

Label coming from a \zlabel inside the note.

```

2018         \tl_set:Ne \l__postnotes_tmpa_tl
2019         {
2020             \zref@extract { \l__postnotes_note_zref_zlabel_str }
2021             { postnotes@tagsup@noteid }
2022         }
2023         \tag_struct_begin:n
2024         {
2025             tag = endnotemark ,
2026             ref = { postnote. \l__postnotes_tmpa_tl } ,
2027         }
2028         \__postnotes_tagsup_store_crossref:nN
2029         { postnote } \l__postnotes_tmpa_tl
2030     }
2031     {
2032         \property_if_recorded:eeTF
2033         { postnote@zlabel@option. \l__postnotes_note_zref_zlabel_str }
2034         { postnotes/tagsup@noteid }
2035         {

```

Label coming from a zlabel option.

```

2036         \tl_set:Ne \l__postnotes_tmpa_tl
2037         {
2038             \property_ref:ee
2039             {
2040                 postnote@zlabel@option.
2041                 \l__postnotes_note_zref_zlabel_str
2042             }
2043             { postnotes/tagsup@noteid }
2044         }
2045         \tag_struct_begin:n
2046         {
2047             tag = endnotemark ,
2048             ref = { postnotemark. \l__postnotes_tmpa_tl } ,
2049         }
2050         \__postnotes_tagsup_store_crossref:nN
2051         { postnotemark } \l__postnotes_tmpa_tl
2052     }
2053     { \tag_struct_begin:n { tag = endnotemark } }
2054 }
2055 \tag_mc_begin:n { }
2056 }
2057 \socket_new_plug:nnn { tagsupport/postnotes/postnotezref/end } { default }
2058 {
2059     \tag_mc_end:
2060     \tag_struct_end: % endnotemark
2061     \tag_mc_begin_pop:n { }
2062 }

```

```

2063         \socket_assign_plug:nn { tagsupport/postnotes/postnotezref/begin } { default }
2064         \socket_assign_plug:nn { tagsupport/postnotes/postnotezref/end } { default }
2065     }
2066 }

```

## 10 Languages

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

```

\pnhdnotes
\pnhdtopage
\pnhdtopages
2067 \tl_new:N \pntitle
2068 \tl_new:N \pnhdnotes
2069 \tl_new:N \pnhdtopage
2070 \tl_new:N \pnhdtopages
2071 \tl_set:Nn \pntitle { Notes }
2072 \tl_set:Nn \pnhdnotes { Notes }
2073 \tl_set:Nn \pnhdtopage { to~page }
2074 \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}
2075 \cs_new_protected:Npn \_postnotes_define_language:nn #1#2
2076 {
2077     \tl_new:c { g__postnotes_language_ #1 _t1 }
2078     \tl_gset:cn { g__postnotes_language_ #1 _t1 } {#2}
2079 }

```

*(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 `\babel language` to execute the setup defined by `\__postnotes_define_language:nn` for `\postnote language` at the `\babel/\language/beforeextras` hook.

```

\__postnotes_set_babel_language:nn {\babel language} {\postnote language}
2080 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
2081 {
2082   \ActivateGenericHook { babel/#1/beforeextras }
2083   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
2084     { g__postnotes_language_ #2 _tl }
2085 }

```

*(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 `\captionslanguage` works as in `babel`'s "old way", meaning it is just a "hook" to which we can append some code. So we use `\captionslanguage` 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 \captionslanguage` hook.

```

\__postnotes_set_polyglossia_language:nn {\polyglossia language}
{\postnote language}
2086 \cs_new_protected:Npn \__postnotes_set_polyglossia_language:nn #1#2
2087 {
2088   \AddToHook { package/polyglossia/after }
2089     {
2090       \exp_args:Nnv \csgappto { captions #1 }
2091         { g__postnotes_language_ #2 _tl }
2092     }
2093 }

```

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

## English

```

2094 \__postnotes_define_language:nn { english }
2095 {
2096   \tl_set:Nn \pntitle      { Notes }
2097   \tl_set:Nn \pnhdnotes   { Notes }
2098   \tl_set:Nn \pnhdtopage  { to~page }
2099   \tl_set:Nn \pnhdtopages { to~pages }

```

```

2100 }
2101 \__postnotes_set_babel_language:nn { english } { english }
2102 \__postnotes_set_babel_language:nn { british } { english }
2103 \__postnotes_set_babel_language:nn { american } { english }
2104 \__postnotes_set_babel_language:nn { canadian } { english }
2105 \__postnotes_set_babel_language:nn { australian } { english }
2106 \__postnotes_set_babel_language:nn { newzealand } { english }
2107 \__postnotes_set_babel_language:nn { UKenglish } { english }
2108 \__postnotes_set_babel_language:nn { USenglish } { english }
2109 \__postnotes_set_polyglossia_language:nn { english } { english }

```

## Portuguese

```

2110 \__postnotes_define_language:nn { portuguese }
2111 {
2112   \tl_set:Nn \pntitle { Notas }
2113   \tl_set:Nn \pnhdnotes { Notas }
2114   \tl_set:Nn \pnhdtopage { da-página }
2115   \tl_set:Nn \pnhdtopages { das-páginas }
2116 }
2117 \__postnotes_set_babel_language:nn { portuguese } { portuguese }
2118 \__postnotes_set_babel_language:nn { brazilian } { portuguese }
2119 \__postnotes_set_babel_language:nn { portuges } { portuguese }
2120 \__postnotes_set_babel_language:nn { brazil } { portuguese }
2121 \__postnotes_set_polyglossia_language:nn { portuguese } { portuguese }

```

## French

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

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

```

2122 \__postnotes_define_language:nn { french }
2123 {
2124   \tl_set:Nn \pntitle { Notes }
2125   \tl_set:Nn \pnhdnotes { Notes }
2126   \tl_set:Nn \pnhdtopage { de-la-page }
2127   \tl_set:Nn \pnhdtopages { des-pages }
2128 }
2129 \__postnotes_set_babel_language:nn { french } { french }
2130 \__postnotes_set_babel_language:nn { acadian } { french }
2131 \__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`.

```

2132 \__postnotes_define_language:nn { german }
2133 {
2134   \tl_set:Nn \pntitle { Endnoten }
2135   \tl_set:Nn \pnhdnotes { Endnoten }
2136   \tl_set:Nn \pnhdtopage { zu-Seite }
2137   \tl_set:Nn \pnhdtopages { zu-Seiten }
2138 }

```

```

2139 \_postnotes_set_babel_language:nn { german }      { german }
2140 \_postnotes_set_babel_language:nn { ngerman }    { german }
2141 \_postnotes_set_babel_language:nn { austrian }   { german }
2142 \_postnotes_set_babel_language:nn { naustrian }  { german }
2143 \_postnotes_set_babel_language:nn { swissgerman } { german }
2144 \_postnotes_set_babel_language:nn { nswissgerman } { german }
2145 \_postnotes_set_polyglossia_language:nn { german } { german }
2146 </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>	
\ActivateGenericHook .....	2082
\addto .....	59
\addtocounter .....	46
\AddToHook .....	110, 131, 359, 459, 1196, 1227, 1270, 1343, 1377, 1379, 1381, 1383, 1390, 1392, 1404, 1409, 1465, 1469, 1476, 1499, 1507, 1537, 1539, 1546, 1549, 1553, 1560, 1572, 1574, 1583, 1586, 1591, 1601, 1604, 1609, 1620, 1622, 1629, 1633, 1881, 1885, 1889, 1892, 1901, 1903, 1913, 1949, 2010, 2083, 2088
\AddToHookNext .....	1371
\AddToHookWithArguments .....	1873
<b>B</b>	
\begin .....	925
\BlockquoteDisable .....	1589
bool commands:	
\bool_do_until:nn .....	1047
\bool_gset_false:N .....	105, 1372
\bool_gset_true:N .....	89, 1347
\bool_if:NTF .....	35, 364, 401, 410, 461, 514, 522, 561, 573, 623, 647, 692, 773, 857, 924, 968, 1000, 1041, 1128, 1165, 1199, 1272, 1327, 1593, 1635, 1732, 1747, 1754, 1762, 1772, 1797, 1875
\bool_if_exist:NTF .....	1631
\bool_lazy_all:nTF .....	614, 1518
\bool_lazy_and:nnTF .....	94, 545, 744, 832, 1024, 1107, 1144, 1176, 1611, 1662
\bool_lazy_or:nnTF .....	541, 1115
\bool_lazy_or_p:nn .....	1151
\bool_new:N .....	88, 224, 332, 333, 334, 388, 427, 434, 435,
445, 452, 577, 580, 604, 605, 606, 783, 1220, 1323, 1536, 1588, 1603, 1872	
\bool_set_false:N .....	231, 341, 350, 351, 366, 610, 611, 612
\bool_set_true:N .....	236, 340, 345, 346, 588, 1538, 1578, 1579, 1580, 1590, 1595, 1596, 1597, 1606, 1615, 1616, 1617, 1624, 1637, 1638, 1639, 1882
\bool_to_str:N .....	53
\bool_until_do:nn .....	862
box commands:	
\box_new:N .....	20
\box_use:N .....	320
\box_wd:N .....	319
<b>C</b>	
\caption .....	16, 17, 40
\chapter .....	192
\citereset .....	42
clist commands:	
\clist_const:Nn .....	1290
\clist_map_inline:Nn .....	1450, 1453, 1456, 1459
\counterwithin .....	17
cs commands:	
\cs_generate_variant:Nn .....	23, 74, 162, 165, 168, 175, 182, 289, 703, 1034, 1322
\cs_if_eq_p:NN .....	1613
\cs_if_exist:NTF .....	39, 62, 171, 178, 188, 1378
\cs_if_exist_p:N .....	548, 1116, 1152, 1153, 1663
\cs_new:Npn .....	21, 80, 176, 277
\cs_new_protected:Npe .....	103
\cs_new_protected:Npn .....	25, 58, 75, 82, 90, 92, 152, 160, 163, 166, 169, 190, 197, 399, 408, 472, 474, 510, 651, 687, 705, 723, 738,

764, 827, 1022, 1036, 1105, 1142, 1163, 1301, 1325, 1424, 1442, 1512, 1933, 1939, 1944, 2075, 2080, 2086	<code>\fp_use:N</code> . . . . . 36
<code>\cs_set:Npn</code> . . . . . 1385, 1386	<b>G</b>
<code>\cs_set_eq:NN</code> . 250, 267, 859, 1268, 1269, 1462, 1489, 1490, 1607, 1625	<code>\gappto</code> . . . . . 60
<code>\cs_set_protected:Npn</code> . . . 1345, 1369	group commands:
<code>\csgappto</code> . . . . . 2090	<code>\group_begin:</code> 205, 512, 629, 639, 740, 766, 829, 885, 926, 930, 1038, 1111, 1148, 1167, 1198, 1303, 1488, 1514
<code>\csundef</code> . . . . . 1451, 1454, 1457, 1460	<code>\group_end:</code> . . . . . . . . . . 214, 574, 633, 645, 755, 780, 904, 996, 1007, 1018, 1103, 1137, 1158, 1193, 1213, 1320, 1493, 1533
<b>D</b>	<b>H</b>
<code>\DeclareInstance</code> . . . . . 1816, 1832, 1834	<code>\hbox</code> . . . . . 284
<code>\DeclareInstanceCopy</code> . . . . . 1828	hbox commands:
<code>\def</code> . . . . . 3	<code>\hbox_set:Nn</code> . . . . . 317
dim commands:	<code>\hspace</code> . . . . . 278, 1836
<code>\dim_compare:nNnTF</code> . . . . . 412	<code>\hyperlink</code> . . . . . 1525
<code>\DocumentMetadata</code> . . . . . 48	<code>\hyperref</code> . . . . . 748
<b>E</b>	<b>I</b>
<code>\EditInstance</code> . . . . . 1830	<code>\IfFormatAtLeastTF</code> . . . . . 5, 6
<code>\end</code> . . . . . 1002	<code>\IfInstanceExistsT</code> . . . . . 1814
<code>\endblockenv</code> . . . . . 1854, 1870	<code>\IfPackageAtLeastTF</code> . . . . . 1551
<code>\endlist</code> . . . . . 259, 276	<code>\IfPackageLoadedTF</code> . . . . . 361
<code>\endnotemark</code> . . . . . 16	<code>\iftoggle</code> . . . . . 1400
<code>\endnotetext</code> . . . . . 16	int commands:
<code>\endrefcontext</code> . . . . . 42	<code>\int_compare:nNnTF</code> . . . . . 1058, 1088
<code>\noteheading</code> . . . . . 28	<code>\int_gadd:Nn</code> . . . . . 98, 475
exp commands:	<code>\int_gincr:N</code> . . . . . . . . . . 517, 767, 768, 830, 1329, 1356
<code>\exp_args:Ne</code> 675, 1330, 1357, 1919, 1927	<code>\int_gset:Nn</code> . . . . . 473
<code>\exp_args:NNe</code> . . . . . 524	<code>\int_incr:N</code> . . . . . 630
<code>\exp_args:NNNo</code> . . . . . 664	<code>\int_new:N</code> . 86, 499, 503, 763, 800, 1324
<code>\exp_args:NNNV</code> . . . . . 632	<code>\int_set:Nn</code> . . . . . 521, 526, 531, 951, 1413, 1416, 1484, 1564, 1567
<code>\exp_args:NNo</code> . . . . . 664	<code>\int_use:N</code> . . . . . 32, 37, 51, 100, 416, 501, 558, 711, 846, 1044, 1333, 1339, 1360, 1366, 1395, 1397, 1444, 1445, 1472, 1556, 1558
<code>\exp_args:Nnv</code> . . . . . 2083, 2090	iow commands:
<code>\exp_args:NV</code> . . . . . . . . . . 657, 660, 727, 732, 925, 1002	<code>\iow_char:N</code> . . . . . 1021, 1217, 1264
<code>\exp_not:N</code> . . . . . 105, 106, 107, 108	<code>\iow_now:Nn</code> 114, 119, 124, 135, 140, 145
<code>\exp_not:n</code> . . . . . 179	<code>\iow_shipout_e:Nn</code> . . . . . 6
<b>F</b>	<code>\item</code> . . . . . 30, 49, 970
<code>\FirstMark</code> . 206, 209, 212, 1275, 1279, 1283, 1287, 1348, 1350, 1352, 1354	<code>\itemindent</code> . . . . . 249, 266
<code>\fmtversion</code> . . . . . 5	<code>\itemsep</code> . . . . . 254, 271
fnote internal commands:	<b>K</b>
<code>\_fnote_gput_ref:nn</code> . . . . . 56	<code>\kern</code> . . . . . 403, 404
<code>\footnote</code> . . . . . 12, 46	keys commands:
<code>\footnotemark</code> . . . . . 12, 15–17	<code>\keys_define:nn</code> . . . . . . . . . . 183, 217, 225, 279, 293,
<code>\footnotesize</code> . . . . . 310	
<code>\footnotetext</code> . . . . . 15, 16, 1386	
<code>\footref</code> . . . . . 56	
fp commands:	
<code>\fp_compare:nNnTF</code> . . . . . 1184	
<code>\fp_new:N</code> . . . . . 578	
<code>\fp_set:Nn</code> . . . . . 587	

	302, 335, 368, 390, 428, 436, 446, 453, 463, 581, 784, 1221, 1229, 1501		
	<code>\keys_set:nn</code> . . . . .	497, 513, 771	
<b>L</b>			
<code>\label</code> . . . . .	17, 50, 54, 57, 727, 1489		
<code>\labelsep</code> . . . . .	278, 1836		
<code>\labelwidth</code> . . . . .	248, 265		
<code>\lastkern</code> . . . . .	13, 413		
<code>\LastMark</code> . . . . .	207, 212, 1277, 1281, 1285, 1289, 1349, 1351, 1353, 1355		
<code>\leftmargin</code> . . . . .	247, 264, 265, 266, 1860, 1861		
<code>\leftskip</code> . . . . .	312		
legacy commands:			
	<code>\legacy_if:nTF</code> . . . . .		
	. . . . .	112, 133, 154, 479, 489, 1576, 1584	
<code>\list</code> . . . . .	245, 262		
<code>\listparindent</code> . . . . .	252, 269		
<code>\lTblrMeasuringBool</code> . . . . .	47, 1631, 1635		
<b>M</b>			
<code>\makeatletter</code> . . . . .	60		
<code>\makeatother</code> . . . . .	60		
<code>\makelabel</code> . . . . .	250, 267		
<code>\MakeLinkTarget</code> . . . . .	2, 556, 959		
mark commands:			
	<code>\mark_insert:nn</code> . . . . .	30, 983, 985, 987, 989	
	<code>\mark_new_class:n</code> . . . . .	816, 817, 818, 819	
<code>\mathchoice</code> . . . . .	16, 46		
<code>\mbox</code> . . . . .	15		
<code>\mkbibendnote</code> . . . . .	40		
mode commands:			
	<code>\mode_if_horizontal:TF</code> . . . . .	708, 719	
	<code>\mode_leave_vertical:</code> . . . . .	707, 975, 978	
msg commands:			
	<code>\msg_error:nnn</code> . . . . .	1247, 1249, 1251, 1253, 1255, 1257, 1259, 1261	
	<code>\msg_line_context:</code> . . . . .		
	. . . . .	383, 1021, 1141, 1162, 1218, 1264	
	<code>\msg_new:nnn</code> . . . . .		
	. . . . .	382, 384, 1020, 1140, 1161, 1215, 1262	
	<code>\msg_warning:nn</code> . . . . .	365, 848, 1157	
	<code>\msg_warning:nnn</code> . . . . .		
	. . . . .	372, 377, 467, 1130, 1210, 1233	
<code>\multfootsep</code> . . . . .	12		
<code>\multiplefootnotemarker</code> . . . . .	12		
<b>N</b>			
<code>\NeedsTeXFormat</code> . . . . .	4		
<code>\newcounter</code> . . . . .	498, 824, 825		
<code>\NewDocumentCommand</code> . . . . .	203, 476, 486, 496, 506, 735, 757, 792, 1509		
<code>\NewDocumentEnvironment</code> . . . . .	243, 260		
<code>\NewHook</code> . . . . .	3, 24, 508, 509, 607, 820, 821, 822, 823		
<code>\newlabel</code> . . . . .	4		
<code>\NewTblrEnviron</code> . . . . .	47		
<code>\nobreak</code> . . . . .	713		
<code>\noindent</code> . . . . .	327		
<code>\normalfont</code> . . . . .	278, 284, 318, 328, 1836		
<b>P</b>			
<code>\PackageError</code> . . . . .	9		
<code>\par</code> . . . . .	31, 232, 1003, 1006		
<code>\parindent</code> . . . . .			
	. . . . .	249, 252, 269, 313, 1845, 1847, 1863	
<code>\parsep</code> . . . . .	253, 270		
<code>\parskip</code> . . . . .	253, 270, 1848, 1864		
<code>\partopsep</code> . . . . .	256, 273, 1851, 1867		
<code>\pnaddtocounteraux</code> . . . . .	14, 472		
<code>\pnhdchapfirst</code> . . . . .	1240, 1250, 1308, 1350		
<code>\pnhdchaplast</code> . . . . .	1241, 1252, 1310, 1351		
<code>\pnhdnamefirst</code> . . . . .	1244, 1258, 1316, 1354		
<code>\pnhdnamelast</code> . . . . .	1245, 1260, 1318, 1355		
<code>\pnhdnotes</code> . . . . .			
	. . . . .	209, 211, 2067, 2097, 2113, 2125, 2135	
<code>\pnhdpagefirst</code> . . . . .	1238, 1246, 1304, 1348		
<code>\pnhdpagelast</code> . . . . .	1239, 1248, 1306, 1349		
<code>\pnhdsectfirst</code> . . . . .	1242, 1254, 1312, 1352		
<code>\pnhdsectlast</code> . . . . .	1243, 1256, 1314, 1353		
<code>\pnhdtopage</code> . . . . .	209, 2067, 2098, 2114, 2126, 2136		
<code>\pnhdtopages</code> . . . . .			
	. . . . .	211, 2067, 2099, 2115, 2127, 2137	
<code>\pnheaderdefault</code> . . . . .	7, 59, 193, 200, 203		
<code>\pnheading</code> . . . . .	7, 185, 188, 850		
<code>\pnidnextnote</code> . . . . .	794, 892		
<code>\pnsetcounteraux</code> . . . . .	14, 472		
<code>\pnthechapter</code> . . . . .	794, 888		
<code>\pnthechapternextnote</code> . . . . .	794, 896		
<code>\pnthepage</code> . . . . .	794, 940		
<code>\pnthesection</code> . . . . .	794, 891		
<code>\pnthesectionnextnote</code> . . . . .	794, 899		
<code>\pntitle</code> . . . . .			
	. . . . .	192, 199, 2067, 2096, 2112, 2124, 2134	
<code>\posnoteref</code> . . . . .	56		
<code>\postnote</code> . . . . .	2, 3, 15, 17, 18, 20, 23, 34, 40, 46, 48–50, 506, 1385		
<code>\postnotemark</code> . . . . .	17		
<code>\postnoteref</code> . . . . .	17, 24, 46, 49, 50, 56, 735		
postnotes commands:			
	<code>\c_postnotes_multi_notemarker_tl</code> . . . . .	387, 403, 404, 413	
	<code>\l_postnotes_note_id_tl</code> . . . . .	18, 499, 525, 534, 538, 539, 549, 556, 557, 569, 769, 772, 776, 779, 1394, 1396, 1398, 1401, 1555, 1557, 1682, 1683, 1686, 1700, 1701, 1704, 1897, 1908	
	<code>\l_postnotes_print_note_id_tl</code> . . . . .	50, 800, 865, 866, 887, 890, 901, 933,	



934, 936, 938, 942, 945, 948, 960,  
963, 1411, 1414, 1417, 1420, 1478,  
1562, 1565, 1738, 1739, 1742, 1778,  
1779, 1782, 1884, 1888, 1915, 1923

postnotes internal commands:

`\l__postnotes_backlink_bool` . . . .  
. . . . . 334, 355, 1026

`\__postnotes_biblatex_citereset_-  
local:` . . . . . 1407, 1442, 1442

`\__postnotes_biblatex_endrefcontext_-  
local:` . . . . . 1406, 1424, 1424

`\l__postnotes_biblatex_orig_-  
refsection_tl` . 43, 1467, 1471, 1474

`\g__postnotes_biblatex_prev_-  
refsection_tl` 1468, 1473, 1482, 1486

`\l__postnotes_check_dupli_bool` . .  
. . . . . 434, 438, 1128

`\__postnotes_check_duplicates:N` .  
. . . . . 34, 853, 1105, 1105

`\__postnotes_check_floats:N` . . . .  
. . . . . 32, 35, 855, 1142, 1142

`\l__postnotes_check_floats_bool` .  
. . . . . 435, 441, 1145

`\l__postnotes_clear_queue_seq` . . .  
. . . . . 800, 843, 1015

`\g__postnotes_countersaux_bool` . . .  
. . . . . 32, 95, 452, 455,  
461, 522, 542, 833, 1041, 1109, 1199

`\g__postnotes_countersaux_prop` . . .  
. . . . . 87, 99, 524

`\l__postnotes_countersaux_step_-  
int` . . . . . 499, 521, 531, 558

`\l__postnotes_csquotes_measuring_-  
bool` . . . . . 1588, 1590, 1593

`\__postnotes_data_name:n` . . . . .  
. . . . . 2, 18, 21, 21, 23, 27, 28, 29,  
31, 33, 42, 46, 48, 50, 52, 54, 60, 61,  
65, 69, 71, 77, 81, 83, 1394, 1396,  
1398, 1401, 1555, 1557, 1897, 1908

`\__postnotes_define_language:nn` .  
. . . . . 59,  
60, 2075, 2075, 2094, 2110, 2122, 2132

`\l__postnotes_deprecated_-  
headervars_bool` . 1220, 1223, 1272

`\__postnotes_extract_pageref:n` . .  
. . . . . 6, 169, 176, 182

`\g__postnotes_firstrun_bool` . . . .  
. . . . . 88, 89, 105, 546, 834, 1108, 1146

`\__postnotes_get_label_if_-  
exist:N` . . . . . 22, 551, 636, 651, 651

`\__postnotes_get_pageref:Nn` . . . .  
. . . . . 6, 169, 169, 175, 932

`\c__postnotes_header_marks_clist`  
. . . . . 1290, 1335, 1362

`\g__postnotes_header_vars_next_-  
bool` . . . . . 1323, 1327, 1347, 1372

`\l__postnotes_hyperlink_bool` 332,  
340, 345, 350, 366, 692, 746, 1025, 1521

`\l__postnotes_hyperref_warn_bool`  
. . . . . 333, 341, 346, 351, 364

`\__postnotes_inhibit_note:` . . . . 608

`\__postnotes_inhibit_note:TF` . . .  
. . . . . 19, 34, 515, 604

`\l__postnotes_inhibit_note_bool` .  
. . . . . 20, 604,  
610, 616, 647, 1578, 1595, 1615, 1637

`\l__postnotes_inside_note_bool` . .  
. . . . . 1872, 1875, 1882

`\g__postnotes_labelseq_seq` . . . . .  
. . . . . 32, 85, 91, 1045, 1046,  
1050, 1075, 1076, 1079, 1101, 1201

`\__postnotes_list_makelabel:n` . . . .  
. . . . . 250, 267, 277

`\__postnotes_make_mark:nnn` . . . . .  
. . . . . 13, 281, 289,  
420, 643, 694, 698, 749, 751, 1527, 1529

`\__postnotes_make_text_mark:nnn` .  
. . . . . 285, 1028, 1032

`\l__postnotes_manual_sortnum_-  
bool` . . . . . 35, 580, 588

`\l__postnotes_mark_tl` . . . . .  
. . . . . 22, 30, 518, 529,  
536, 540, 576, 583, 621, 638, 734, 1506

`\l__postnotes_mark_typeset_tl` . . .  
. . . . . 499,  
540, 551, 569, 626, 634, 636, 638, 643

`\l__postnotes_maybe_multi_bool` . .  
. . . . . 34, 53, 445, 448, 543, 601

`\l__postnotes_multiple_bool` . . . .  
. . . . . 388, 392, 401, 410

`\__postnotes_multiple_check:` . . .  
. . . . . 13, 408, 712

`\__postnotes_multiple_prepare:` . .  
. . . . . 12, 399, 718

`\l__postnotes_multisep_tl` . . . . .  
. . . . . 386, 395, 420

`\l__postnotes_nomark_bool` . . . . .  
. . . . . 514, 561, 573, 577, 597, 618

`\__postnotes_note:nn` . . . . .  
. . . . . 18, 22, 23, 507, 508, 510

`\g__postnotes_note_id_int` . . . . .  
. . . . . 18, 32, 499, 517, 768, 1061, 1091

`\l__postnotes_note_label_str` . . .  
. . . . . 579, 599,  
653, 676, 682, 725, 727, 728, 1894, 1898

`\__postnotes_note_ref:nn` . . . . .  
. . . . . 24, 736, 737, 738

<code>\l__postnotes_note_ref_label_str</code>	<code>\g__postnotes_print_queue_seq</code>
..... 737, 741, 1962, 1968, 1981, 1987	34, 35, 800, 836, 840, 843, 847, 853,
<code>\l__postnotes_note_set_labels_tl</code>	854, 855, 862, 864, 870, 876, 910, 916
..... 499, 553, 564, 570	<code>\l__postnotes_print_section_tl</code> ..
<code>\l__postnotes_note_zlabel_str</code> ...	..... 800, 937, 988
..... 44, 655, 657, 661,	<code>\l__postnotes_print_sectname_tl</code> .
667, 731, 732, 1498, 1503, 1905, 1909	..... 800, 939, 990
<code>\__postnotes_note_zref:nn</code> .....	<code>\l__postnotes_print_type_curr_tl</code>
..... 44, 1510, 1511, 1512	..... 800, 867, 868, 906, 1011
<code>\l__postnotes_note_zref_zlabel_-</code>	<code>\l__postnotes_print_type_next_tl</code>
str 1511, 1515, 2015, 2020, 2033, 2041	.... 800, 873, 880, 882, 913, 920, 997
<code>\l__postnotes_post_printnote_tl</code> .	<code>\l__postnotes_print_type_prev_tl</code>
..... 232, 292, 299, 995	..... 800, 852, 905, 922, 1010
<code>\l__postnotes_post_textmark_tl</code> ..	<code>\l__postnotes_print_typeset_-</code>
..... 291, 297, 965	mark_tl .....
<code>\g__postnotes_postnote_counteraux_-</code>	..... 30, 800, 955, 973, 980
int .....	<code>\__postnotes_prop_gclear:n</code> .....
86, 98, 100, 473, 475	..... 4, 75, 82, 1016
<code>\l__postnotes_pre_textmark_tl</code> ...	<code>\__postnotes_prop_get:nnN</code> .....
..... 290, 295, 961	.... 4, 75, 75, 866, 878, 886, 889,
<code>\l__postnotes_print_as_list_bool</code>	894, 897, 900, 918, 934, 936, 938,
224, 231, 236, 857, 924, 968, 1000,	941, 944, 947, 1170, 1171, 1174,
1732, 1747, 1754, 1762, 1772, 1797	1175, 1180, 1182, 1411, 1414, 1417,
<code>\l__postnotes_print_chapter_tl</code> ..	1420, 1478, 1562, 1565, 1915, 1923
..... 800, 935, 986	<code>\__postnotes_prop_item:nn</code> .....
<code>\l__postnotes_print_content_tl</code> ..	.... 4, 75, 80, 1119, 1124, 1131, 1206
..... 800, 902, 903, 949, 992	<code>\g__postnotes_queue_seq</code> .....
<code>\l__postnotes_print_counter_tl</code> ..	.... 18, 499, 533, 769, 841, 844, 1205
..... 800, 946, 952	<code>\c__postnotes_ref_prefix_tl</code> .....
<code>\l__postnotes_print_env_tl</code> .....	..... 4, 84, 108, 171,
..... 223, 233, 237, 925, 1002	172, 178, 179, 549, 1116, 1152, 1153
<code>\l__postnotes_print_format_tl</code> ...	<code>\l__postnotes_restore_tmp_tl</code> ...
..... 216, 219, 928	... 1376, 1412, 1413, 1415, 1416,
<code>\g__postnotes_print_header_vars_-</code>	1418, 1419, 1421, 1422, 1479, 1481,
int .....	1485, 1487, 1563, 1564, 1566, 1567,
1329, 1333, 1339, 1356, 1360, 1366	1916, 1917, 1920, 1924, 1925, 1928
<code>\g__postnotes_print_labelseq_-</code>	<code>\l__postnotes_saved_spacefactor_-</code>
queue_seq 837, 1035, 1072, 1099, 1156	multi_tl .....
<code>\l__postnotes_print_mark_tl</code> ....	..... 389, 415, 422
..... 800, 943, 954, 964	<code>\l__postnotes_saved_spacefactor_-</code>
<code>\l__postnotes_print_note_id_-</code>	tl .....
next_tl .....	..... 704, 710, 720
800, 872,	<code>\g__postnotes_sectid_int</code> 51, 763, 767
877, 879, 893, 895, 898, 912, 917, 919	<code>\__postnotes_section:nn</code> .....
<code>\__postnotes_print_notes:</code> .....	..... 25, 760, 763, 764
..... 26, 27, 31, 35, 793, 827, 827	<code>\l__postnotes_section_exp_bool</code> ..
<code>\l__postnotes_print_page_tl</code> ....	..... 773, 783, 788
..... 800, 932, 940, 984	<code>\g__postnotes_section_name_tl</code> ...
<code>\l__postnotes_print_plain_mark_-</code>	..... 49, 770, 782, 786
bool .....	<code>\__postnotes_set_babel_language:nn</code>
20,	..... 60, 2080, 2080, 2101, 2102,
605, 611, 617, 1579, 1596, 1616, 1638	2103, 2104, 2105, 2106, 2107, 2108,
<code>\l__postnotes_print_plain_mark_-</code>	2117, 2118, 2119, 2120, 2129, 2130,
stepcounter_bool .....	2139, 2140, 2141, 2142, 2143, 2144
20, 47,	<code>\__postnotes_set_header_vars:n</code> ..
606, 612, 623, 1580, 1597, 1617, 1639	..... 1301, 1322, 1336, 1363
<code>\g__postnotes_print_postnotes_-</code>	
int .....	
800, 830, 846, 1044	

<code>\__postnotes_set_header_vars_-</code>	<code>\l__postnotes_tmpa_tl</code>
bool: . . . . . 1014, 1269, 1369	. . . . . <a href="#">16</a> , 41, 43, 45, 47, 55, 64, 66, 68, 70, 72, 206, 208, 525, 526, 527, 631, 634, 775, 777, 1043, 1045, 1046, 1048, 1170, 1172, 1174, 1177, 1181, 1185, 1965, 1974, 1977, 1984, 1993, 1996, 2018, 2026, 2029, 2036, 2048, 2051
<code>\__postnotes_set_header_vars_-</code>	<code>\l__postnotes_tmpb_seq</code>
first: . . . . . 856, 1268, 1345	. . . . . <a href="#">16</a> , 1040, 1063, 1074, 1092, 1101
<code>\__postnotes_set_headers_vars_-</code>	<code>\l__postnotes_tmpb_tl</code>
next: . . . . . 1325, 1344	<a href="#">16</a> , 207, 208, 1048, 1050, 1052, 1059, 1064, 1068, 1171, 1172, 1175, 1178, 1183, 1185
<code>\__postnotes_set_label:nnnn</code>	<code>\__postnotes_typeset_mark:nnN</code>
. . . . . <a href="#">6</a> , <a href="#">152</a> , 152, 161, 164, 167	. . . . . <a href="#">22</a> , 568, <a href="#">687</a> , <a href="#">687</a> , 703
<code>\__postnotes_set_mark_page_-</code>	<code>\__postnotes_typeset_mark_-</code>
label:nn . . . . . <a href="#">6</a> , <a href="#">152</a> , 160, 162, 557	wrapper:nnn . . . . .
<code>\__postnotes_set_polyglossia_-</code>	. . . . . <a href="#">22</a> , 642, <a href="#">687</a> , 689, 705, 742, 1516
language:nn . . . . . <a href="#">60</a> , <a href="#">2086</a> , 2086, 2109, 2121, 2131, 2145	<code>\__postnotes_typeset_text_-</code>
<code>\__postnotes_set_print_label:n</code>	mark:nn . . . . . <a href="#">31</a> , 962, <a href="#">1022</a> , 1022, 1034
. . . . . <a href="#">6</a> , <a href="#">152</a> , 166, 168, 845	<code>\l__postnotes_zrefhyperref_bool</code>
<code>\__postnotes_set_section_page_-</code>	. . . . . 1522, 1536, 1538
label:n . . . . . <a href="#">6</a> , <a href="#">152</a> , 163, 165, 772	<code>\postnotessection</code>
<code>\__postnotes_set_user_labels:</code>	. . . . . <a href="#">3</a> , <a href="#">24</a> , <a href="#">25</a> , <a href="#">757</a>
. . . . . <a href="#">44</a> , 559, <a href="#">723</a> , 723	<code>\postnotesetup</code>
<code>\l__postnotes_sort_bool</code>	. . . . . <a href="#">15</a> , 462, <a href="#">496</a> , 1378, 1584
. . . . . 427, 430, 1165	<code>\postnotetext</code>
<code>\l__postnotes_sort_num_fp</code>	. . . . . 17
. . . . . 36, 578, 587	<code>\postnoterezref</code>
<code>\__postnotes_sort_queue:N</code>	. . . . . <a href="#">44</a> , <a href="#">50</a> , <a href="#">57</a> , <a href="#">1509</a>
. . . . . 35, 854, <a href="#">1163</a> , 1163	prg commands:
<code>\__postnotes_split_labelseq:</code>	<code>\prg_do_nothing:</code>
. . . . . 831, <a href="#">1035</a> , 1036	. . . . . 1268, 1269
<code>\__postnotes_step_countersaux:nnn</code>	<code>\prg_new_protected_conditional:Npnn</code>
. . . . . <a href="#">4</a> , <a href="#">84</a> , 92, 107	. . . . . 608
<code>\__postnotes_store:nn</code>	<code>\prg_return_false:</code>
. . . . . <a href="#">3</a> , <a href="#">24</a> , 25, 539	. . . . . 649
<code>\__postnotes_store_labelseq:nn</code>	<code>\prg_return_true:</code>
. . . . . <a href="#">4</a> , <a href="#">84</a> , 90, 106	. . . . . 648
<code>\__postnotes_store_section:nn</code>	<code>\printpostnotes</code>
. . . . . <a href="#">3</a> , <a href="#">58</a> , 58, 74, 776, 779	. . . . . <a href="#">17</a> , <a href="#">18</a> , <a href="#">25-27</a> , <a href="#">31-34</a> , <a href="#">41</a> , <a href="#">42</a> , <a href="#">49-51</a> , <a href="#">56</a> , <a href="#">792</a>
<code>\l__postnotes_tabularx_inside_-</code>	prop commands:
env_bool . . . . . 1603, 1606, 1612, 1624	<code>\prop_gclear:N</code>
<code>\__postnotes_tabularx_saved_-</code>	. . . . . 83
write:Nn . . . . . 1607, 1613, 1625	<code>\prop_get:NnNTF</code>
<code>\g__postnotes_tagsup_crossrefs_-</code>	. . . . . 77
prop . . . . . 1938, 1941, 1951	<code>\prop_gpop:NnNTF</code>
<code>\__postnotes_tagsup_gput_ref:nn</code>	. . . . . 524
. . . . . <a href="#">56</a> , <a href="#">1944</a> , 1944, 1953	<code>\prop_gput:Nnn</code>
<code>\__postnotes_tagsup_store_-</code>	. . . . . 28, 29, 31, 33, 42, 46, 48, 50, 52, 54, 61, 65, 69, 71, 99, 1394, 1396, 1398, 1401, 1555, 1557, 1896, 1907, 1935, 1941
crossref:nN . . . . . <a href="#">56</a> , <a href="#">1932</a> , 1939, 1976, 1995, 2028, 2050	<code>\prop_item:Nn</code>
<code>\__postnotes_tagsup_store_-</code>	. . . . . 81, 1954
sstructnum:nN . . . . . <a href="#">56</a> , 1685, 1703, 1741, 1781, <a href="#">1932</a> , 1933	<code>\prop_map_inline:Nn</code>
<code>\g__postnotes_tagsup_structnums_-</code>	. . . . . 1951
prop . . . . . 1932, 1935, 1954	<code>\prop_new:N</code>
<code>\l__postnotes_tmpa_box</code>	. . . . . 27, 60, 87, 1932, 1938
. . . . . <a href="#">16</a> , 317, 319, 320	property commands:
<code>\l__postnotes_tmpa_seq</code>	<code>\property_if_recorded:nnTF</code>
. . . . . <a href="#">16</a> , 1039, 1067, 1073, 1074, 1076, 1094, 1100, 1112, 1121, 1127, 1136, 1149, 1156, 1201, 1205, 1208, 1211	. . . . . 675, 1961, 1980, 2032
	<code>\property_new:nnnn</code>
	. . . . . 734, 1274, 1276, 1278, 1280, 1282, 1284, 1286, 1288, 1883
	<code>\property_record:nN</code>
	. . . . . 1330, 1357
	<code>\property_record:nn</code>
	. . . . . 728, 1877, 1919, 1927

`\property_ref:nn` ..... 681, 1305, 1307, 1309, 1311, 1313, 1315, 1317, 1319, 1967, 1986, 2038  
`\providecommand` ..... 5, 116, 121, 126, 137, 142, 147  
`\ProvidesExplPackage` ..... 14

### R

`\ref` ..... 2, 749, 751  
`\refsection` ..... 43, 1492  
`\refstepcounter` ..... 17  
`\RenewDocumentEnvironment` ... 1839, 1855  
`\rightmargin` ..... 251, 268  
`\rightskip` ..... 311

### S

scan commands:  
`\scan_stop:` ..... 405, 423, 721  
`\section` ..... 199

seq commands:  
`\seq_clear:N` ..... 1039, 1040, 1112  
`\seq_concat:NNN` ..... 1074  
`\seq_count:N` ..... 1211  
`\seq_gclear:N` ..... 844  
`\seq_get_left:NN` ..... 876, 916  
`\seq_gpop_left:NN` ..... 864, 1050  
`\seq_gput_right:Nn` . 91, 533, 769, 1046  
`\seq_gset_eq:NN` ..... 836, 840, 1072, 1076, 1099, 1101, 1136  
`\seq_gsort:Nn` ..... 1168  
`\seq_if_empty:NTF` . 847, 870, 910, 1208  
`\seq_if_empty_p:N` ..... 862  
`\seq_if_eq:NNTF` ..... 35  
`\seq_if_in:NnTF` ..... 1045  
`\seq_map_inline:Nn` .. 1015, 1079, 1113  
`\seq_new:N` 18, 19, 85, 502, 801, 815, 1035  
`\seq_put_right:Nn` ..... 1063, 1067, 1092, 1094, 1121, 1127  
`\seq_set_eq:NN` ..... 843  
`\seq_set_filter:NNn` . 1149, 1201, 1205  
`\setcounter` ..... 46, 826  
`\setlength` 247, 248, 249, 251, 252, 253, 254, 255, 256, 264, 265, 266, 268, 269, 270, 271, 272, 273, 311, 312, 313  
`\setlocalecaption` ..... 59

skip commands:  
`\skip_horizontal:n` ..... 319  
`\small` ..... 220

socket commands:  
`\socket_assign_plug:nn` .. 640, 641, 1708, 1709, 1710, 1711, 1722, 1723, 1728, 1729, 1750, 1751, 1768, 1769, 1812, 1813, 2008, 2009, 2063, 2064  
`\socket_new:nn` ..... 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661  
`\socket_new_plug:nnn` ..... 1676, 1689, 1695, 1706, 1712, 1717, 1724, 1726, 1730, 1745, 1752, 1760, 1770, 1795, 1958, 2002, 2012, 2057

sort commands:  
`\sort_return_same:` .. 1187, 1189, 1191  
`\sort_return_swapped:` ..... 1186  
`\space` ..... 11  
`\spacefactor` ..... 416, 422, 711, 720  
`\stepcounter` ..... 17, 520, 625, 884

str commands:  
`\str_case:nnTF` ..... 33, 1051, 1081  
`\str_if_empty:NTF` ..... 653, 655, 725, 731, 1894, 1905  
`\str_if_eq:nnTF` ..... 1123  
`\str_if_eq_p:nn` ..... 96, 1118, 1177, 1178, 1202, 1206  
`\str_new:N` ..... 579, 737, 1498, 1511  
`\str_set:Nn` ..... 741, 1515

### T

tag commands:  
`\tag_get:n` ..... 1936, 1942  
`\tag_if_active:TF` ..... 1946  
`\tag_if_active_p:` ..... 1663, 1664  
`\tag_mc_begin:n` ..... 1687, 1715, 1757, 1786, 1792, 2000, 2055  
`\tag_mc_begin_pop:n` ..... 1693, 1720, 2006, 2061  
`\tag_mc_end:` ..... 1691, 1719, 1764, 1799, 1806, 2004, 2059  
`\tag_mc_end_push:` ..... 1678, 1714, 1960, 2014  
`\tag_socket_use:n` 419, 421, 563, 565, 700, 701, 753, 754, 927, 957, 966, 972, 979, 991, 993, 994, 999, 1531, 1532  
`\tag_struct_begin:n` ..... 56, 1679, 1697, 1734, 1756, 1774, 1783, 1784, 1789, 1790, 1971, 1990, 1998, 2023, 2045, 2053  
`\tag_struct_end:` ..... 1692, 1707, 1748, 1765, 1801, 1802, 1803, 1808, 1809, 2005, 2060  
`\tag_struct_gput:nnn` ..... 1947  
`\tag_struct_object_ref:n` ..... 1947  
`\tag_tool:n` 1725, 1785, 1791, 1800, 1807  
`\tagpdfsetup` ..... 1666

$\TeX$  and  $\LaTeX$  2 $\epsilon$  commands:  
`\@afterindentfalse` ..... 859  
`\@afterindenttrue` ..... 28, 859, 860

<code>\@auxout</code>	156, 481, 491	<code>\hyper@linkend</code>	696, 1030
<code>\@bsphack</code>	478, 488, 514, 759	<code>\hyper@linkstart</code>	695, 1029
<code>\@capttype</code>	1378	<code>\ifmeasuring@</code>	46
<code>\@currentHref</code>	537	<code>\p@postnote</code>	536, 954
<code>\@currentcounter</code>	535, 950	<code>\post@note</code>	4, 5, 84, 117, 138, 157
<code>\@currentlabel</code>	536, 953	<code>\postnote@addtocounteraux</code>	5, 14, 127, 148, 472
<code>\@esphack</code>	484, 494, 573, 761	<code>\postnote@setcounteraux</code>	5, 14, 122, 143, 472
<code>\@footnotemark</code>	22	<code>\postnotes@required@kernel</code>	3, 4, 6, 11
<code>\@ifl@t@r</code>	5	<code>\protected@edef</code>	41, 45, 64, 68, 206, 207, 529, 536, 626, 631, 679, 775, 953
<code>\@mainaux</code>	114, 119, 124, 135, 140, 145	<code>\protected@write</code>	156, 481, 491
<code>\@makecaption</code>	40	<code>\protected@xdef</code>	1304, 1306, 1308, 1310, 1312, 1314, 1316, 1318
<code>\@makefnmark</code>	9	<code>\z@</code>	1462
<code>\@mkboth</code>	193, 200	<code>\zref@extract</code>	666, 2020
<code>\@newl@bel</code>	4, 108	<code>\zref@extractdefault</code>	1526
<code>\@textsuperscript</code>	284, 318	<code>\zref@ifrefcontainsprop</code>	660, 2015
<code>\blx@bibfiles</code>	43	<code>\zref@ifrefundefined</code>	657
<code>\blx@edef@refcontext</code>	1422, 1433	<code>\zref@localaddprop</code>	1508, 1890
<code>\blx@endrefsection</code>	1491	<code>\zref@newprop</code>	1506, 1887
<code>\blx@err@endnote</code>	1386	<code>\text</code>	16, 17, 46
<code>\blx@info</code>	1490	<code>\textnormal</code>	46
<code>\blx@lasthash@foot</code>	1449	<code>\textsuperscript</code>	12, 13
<code>\blx@lasthash@text</code>	1448	<code>\textup</code>	46
<code>\blx@lastkey@foot</code>	1447	<code>\thechapter</code>	41, 64
<code>\blx@lastkey@text</code>	1446	<code>\theHpostnote</code>	17
<code>\blx@lastmpfn</code>	1462	<code>\thepage</code>	161, 164
<code>\blx@refcontext@context</code>	1402	<code>\thepostnote</code>	21, 529, 626, 631
<code>\blx@refcontext@labelalphanametemplatenam</code>	1432, 1439	<code>\thesection</code>	45, 68
<code>\blx@refcontext@labelprefix</code>	1427	<code>\the commands:</code>	
<code>\blx@refcontext@labelprefix@real</code>	1428	<code>\c_empty_tl</code>	180
<code>\blx@refcontext@sortingnamekeytemplatenam</code>	1430, 1436	<code>\tl_clear:N</code>	55, 72, 78, 173, 527, 1427, 1428, 1444, 1445, 1452, 1455, 1458, 1461
<code>\blx@refcontext@sortingtemplatenam</code>	1429, 1435	<code>\tl_const:Nn</code>	84, 387
<code>\blx@refcontext@uniquenametemplatenam</code>	1431, 1438	<code>\tl_gclear:N</code>	770
<code>\blx@sorting</code>	1429	<code>\tl_gput_right:Nn</code>	60
<code>\blx@theendnote</code>	1385	<code>\tl_gset:Nn</code>	537, 1246, 1248, 1250, 1252, 1254, 1256, 1258, 1260, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1473, 2078
<code>\blx@theendnotetext</code>	1386	<code>\tl_gset_eq:NN</code>	1486
<code>\blx@trackhash@foot</code>	1453, 1455	<code>\tl_if_empty:NTF</code>	518, 621, 1917, 1925
<code>\blx@trackhash@text</code>	1450, 1452	<code>\tl_if_eq:NNTF</code>	208, 1155, 1172, 1480
<code>\blx@trackkeys@foot</code>	1459, 1461	<code>\tl_if_eq:NnTF</code>	868, 882, 922, 997
<code>\blx@trackkeys@text</code>	1456, 1458	<code>\tl_if_eq:nnTF</code>	229
<code>\c@blx@maxsection</code>	1484	<code>\tl_if_eq_p:NN</code>	1048
<code>\c@postnote</code>	21, 32, 37, 526, 630	<code>\tl_item:Nn</code>	1052, 1059, 1068
<code>\c@postnotetext</code>	951	<code>\tl_item:nn</code>	1082, 1089, 1095, 1202
<code>\c@refsection</code>	1395, 1413, 1444, 1445, 1472	<code>\tl_new:N</code>	16, 17, 216, 223, 290, 291, 292, 386, 389, 500, 504, 505, 576, 704, 782, 794, 795, 796,
<code>\c@refsegment</code>	1397, 1416		
<code>\c@zc@abschap</code>	1556, 1564		
<code>\c@zc@abssec</code>	1558, 1567		
<code>\FN@mf@check</code>	12		
<code>\FN@mf@prepare</code>	12		

797, 798, 799, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1376, 1467, 1468, 2067, 2068, 2069, 2070, 2077	. 116, 117, 121, 122, 126, 127, 137, 138, 142, 143, 147, 148, 157, 482, 492
<code>\tl_set:Nn</code> .....	<code>\topsep</code> ..... 255, 272, 1850, 1866
. 172, 232, 233, 237, 415, 501, 535, 553, 634, 664, 710, 852, 872, 873, 892, 905, 912, 913, 950, 955, 1010, 1043, 1429, 1430, 1431, 1432, 1471, 1965, 1984, 2018, 2036, 2071, 2072, 2073, 2074, 2096, 2097, 2098, 2099, 2112, 2113, 2114, 2115, 2124, 2125, 2126, 2127, 2134, 2135, 2136, 2137	<b>U</b>
<code>\tl_set_eq:NN</code> ..... 540, 638, 940	<code>\undef</code> ..... 1446, 1447, 1448, 1449
<code>\togglefalse</code> ..... 1426	<code>\unkern</code> ..... 417, 418
<code>\toggletrue</code> ..... 1382	use commands:
token commands:	<code>\use:N</code> ..... 1419
<code>\token_to_str:N</code> .....	<code>\use_none:n</code> ..... 1489, 1490
	<code>\UseHook</code> 56, 532, 555, 613, 851, 931, 958, 982
	<code>\UseInstance</code> ..... 1841, 1857
	<b>W</b>
	<code>\write</code> ..... 34, 1607, 1613, 1625
	<b>Z</b>
	<code>\zcsetup</code> ..... 1541, 1547
	<code>\zlabel</code> ..... 50, 54, 58, 732
	<code>\zref</code> ..... 1527, 1529