

The `postnotes` package

Code documentation

`gusbrs`

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

Version v0.3.0 – 2024-10-15

Contents

1	Initial setup	2
2	Data	2
3	Options	6
4	<code>\postnote</code>	13
5	<code>\postnoteref</code>	19
6	<code>\postnotesection</code>	20
7	<code>\printpostnotes</code>	21
8	Headers	27
9	Compatibility	34
10	Languages	53
	Index	56

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 (`ltnnews34`, `ltfilehook`). Furthermore, the kernel of 2022-06-01 introduced a couple of very nice features which simplifies the relation with `hyperref` (`ltnnews35`, `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} {2024-10-15} {0.3.0}
15 {Endnotes for LaTeX}
```

`\l__postnotes_tmpa_tl` Temporary scratch variables.

```
\l__postnotes_tmpb_tl 16 \tl_new:N \l__postnotes_tmpa_tl
\l__postnotes_tmpa_seq 17 \tl_new:N \l__postnotes_tmpb_tl
\l__postnotes_tmpa_box 18 \seq_new:N \l__postnotes_tmpa_seq
19 \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>}
20 \cs_new:Npn \__postnotes_data_name:n #1
21   { g__postnotes_#1_data_prop }
22 \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 $\rangle$ }
23 \NewHook { postnotes/note/store }
24 \cs_new_protected:Npn \__postnotes_store:nn #1#2
25 {
26   \prop_new:c { \__postnotes_data_name:e {#1} }
27   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { note }
28   \prop_gput:cne { \__postnotes_data_name:e {#1} } { mark }
29     { \l__postnotes_mark_tl }
30   \prop_gput:cne { \__postnotes_data_name:e {#1} } { counter }
31     { \int_use:N \c@postnote }
32   \prop_gput:cne { \__postnotes_data_name:e {#1} } { sortnum }
33     {
34       \bool_if:NTF \l__postnotes_manual_sortnum_bool
35         { \fp_use:N \l__postnotes_sort_num_fp }
36         { \int_use:N \c@postnote }
37     }
38   \cs_if_exist:cT { chapter }
39     {
40       \prop_gput:cne { \__postnotes_data_name:e {#1} }
41         { thechapter } { \thechapter }
42     }
43   \prop_gput:cne { \__postnotes_data_name:e {#1} } { thesection }
44     { \thesection }
45   \prop_gput:cne { \__postnotes_data_name:e {#1} } { pnsectname }
46     { \g__postnotes_section_name_tl }
47   \prop_gput:cne { \__postnotes_data_name:e {#1} } { pnsectid }
48     { \int_use:N \g__postnotes_sectid_int }
49   \prop_gput:cne { \__postnotes_data_name:e {#1} } { multibool }
50     { \bool_to_str:N \l__postnotes_maybe_multi_bool }
51   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}
52   \UseHook { postnotes/note/store }
53 }
```

(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 $\rangle$ }
54 \cs_new_protected:Npn \__postnotes_store_section:nn #1#2
55 {
56   \prop_new:c { \__postnotes_data_name:e {#1} }
```

```

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

```

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

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

`__postnotes_prop_item:nn`
`__postnotes_prop_gclear:n`

```

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

68 \cs_new_protected:Npn \__postnotes_prop_get:nnN #1#2#3
69 {
70   \prop_get:cnNF { \__postnotes_data_name:e {#1} } {#2} #3
71   { \tl_clear:N #3 }
72 }
73 \cs_new:Npn \__postnotes_prop_item:nn #1#2
74 { \prop_item:cn { \__postnotes_data_name:e {#1} } {#2} }
75 \cs_new_protected:Npn \__postnotes_prop_gclear:n #1
76 { \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` The `\newlabel` equivalent for postnotes. Based on the kernel's `\@newl@bel` so that we get L^AT_EX checks for multiple and changed references for free (procedure learnt from `zref`). `\post@note`, when the `.aux` file is read, defines macros named `\postnote@r@{label name}`, according to the prefix set by `\c__postnotes_ref_prefix_tl`.

```

\post@note {<label name>} {<label content>}

77 \tl_const:Nn \c__postnotes_ref_prefix_tl { postnote@r }
78 \cs_new_protected:Npe \post@note #1#2
79 { \exp_not:N \@newl@bel { \c__postnotes_ref_prefix_tl } {#1} {#2} }

```

(End of definition for `\post@note`.)

And ensure `\post@note` is defined in the `.aux` file. The hooks are the same used by `hyperref` for similar purpose.

```

80 \AddToHook { begindocument }
81 {
82   \legacy_if:nT { @filesw }
83   {
84     \iow_now:Ne \@mainaux
85     { \token_to_str:N \providecommand \token_to_str:N \post@note [2]{ }
86     }

```

```

87 }
88 \AddToHook { include/before }
89 {
90   \legacy_if:nT { @filesw }
91   {
92     \iow_now:Ne \@partaux
93     { \token_to_str:N \providecommand \token_to_str:N \post@note [2]{} }
94   }
95 }

```

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

```

\__postnotes_set_mark_page_label:n
\__postnotes_set_text_page_label:n
\__postnotes_set_print_page_label:n

\__postnotes_set_label:nn {<label name>} {<value>}
\__postnotes_set_mark_page_label:n {<note id>}
\__postnotes_set_text_page_label:n {<note id>}
\__postnotes_set_print_page_label:n {<note id>}

96 \cs_new_protected:Npn \__postnotes_set_label:nn #1#2
97 {
98   \legacy_if:nT { @filesw }
99   {
100     \iow_shipout_x:Nn \@auxout
101     { \token_to_str:N \post@note { #1 } { #2 } }
102   }
103 }
104 \cs_new_protected:Npn \__postnotes_set_mark_page_label:n #1
105 { \__postnotes_set_label:nn { mark@ #1 } { \thepage } }
106 \cs_generate_variant:Nn \__postnotes_set_mark_page_label:n { e }
107 \cs_new_protected:Npn \__postnotes_set_text_page_label:n #1
108 { \__postnotes_set_label:nn { text@ #1 } { \int_use:N \c@page } }
109 \cs_generate_variant:Nn \__postnotes_set_text_page_label:n { e }
110 \cs_new_protected:Npn \__postnotes_set_print_page_label:n #1
111 { \__postnotes_set_label:nn { print@ #1 } { \int_use:N \c@page } }
112 \cs_generate_variant:Nn \__postnotes_set_print_page_label:n { e }

```

(End of definition for `__postnotes_set_label:nn` 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>}

113 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
114 {
115   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
116   { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
117   { \tl_clear:N #1 }
118 }
119 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Ne }
120 \cs_new:Npn \__postnotes_extract_pageref:n #1
121 {
122   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
123   { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
124   { \c_empty_tl }

```

```

125 }
126 \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

```

127 \keys_define:nn { postnotes/setup }
128 {
129   heading .cs_set_protected:Np = \pnheading ,
130   heading .value_required:n = true ,
131 }

```

\pnheading Provide default value for \pnheading.

```

132 \cs_if_exist:cTF { chapter }
133 {
134   \cs_new_protected:Npn \pnheading
135   {
136     \chapter*{\pntitle}
137     \@mkboth{\pnheaderdefault}{\pnheaderdefault}
138   }
139 }
140 {
141   \cs_new_protected:Npn \pnheading
142   {
143     \section*{\pntitle}
144     \@mkboth{\pnheaderdefault}{\pnheaderdefault}
145   }
146 }

```

(End of definition for \pnheading.)

format option

```

147 \tl_new:N \l__postnotes_print_format_tl
148 \keys_define:nn { postnotes/setup }
149 {
150   format .tl_set:N = \l__postnotes_print_format_tl ,
151   format .initial:n = { \small } ,
152   format .value_required:n = true ,
153 }

```

listenv option

```

154 \tl_new:N \l__postnotes_print_env_tl
155 \bool_new:N \l__postnotes_print_as_list_bool
156 \keys_define:nn { postnotes/setup }
157 {
158   listenv .code:n =
159   {
160     \tl_if_eq:nnTF {#1} { none }
161     {

```

```

162         \bool_set_false:N \l__postnotes_print_as_list_bool
163         \tl_set:Nn \l__postnotes_post_printnote_tl { \par }

```

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

```

164         \tl_set:Nn \l__postnotes_print_env_tl { itemize }
165     }
166     {
167         \bool_set_true:N \l__postnotes_print_as_list_bool
168         \tl_set:Nn \l__postnotes_print_env_tl {#1}
169     }
170 },
171 listenv .initial:n = { postnoteslist } ,
172 listenv .value_required:n = true ,
173 }

```

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

```

174 \NewDocumentEnvironment { postnoteslist } { }
175 {
176     \list { }
177     {
178         \setlength { \leftmargin } { Opt }
179         \setlength { \labelwidth } { Opt }
180         \setlength { \itemindent } { .5\parindent }
181         \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n
182         \setlength { \rightmargin } { Opt }
183         \setlength { \listparindent } { \parindent }
184         \setlength { \parsep } { \parskip }
185         \setlength { \itemsep } { Opt }
186         \setlength { \topsep } { .5\topsep }
187         \setlength { \partopsep } { .5\partopsep }
188     }
189 }
190 { \endlist }
191 \NewDocumentEnvironment { postnoteslisthang } { }
192 {
193     \list { }
194     {
195         \setlength { \leftmargin } { 1em }
196         \setlength { \labelwidth } { -\leftmargin }
197         \setlength { \itemindent } { -2\leftmargin }
198         \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n
199         \setlength { \rightmargin } { Opt }
200         \setlength { \listparindent } { \parindent }
201         \setlength { \parsep } { \parskip }
202         \setlength { \itemsep } { Opt }
203         \setlength { \topsep } { .5\topsep }
204         \setlength { \partopsep } { .5\partopsep }
205     }
206 }
207 { \endlist }
208 \cs_new:Npn \l__postnotes_list_makelabel:n #1
209 { \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.

```
210 \keys_define:nn { postnotes/setup }
211   {
212     makemark .cs_set:Np = \__postnotes_make_mark:nnn #1#2#3 ,
213     makemark .value_required:n = true ,
```

From the default kernel definition of \@makefnmark.

```
214     makemark .initial:n =
215       { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
216     maketextmark .cs_set:Np = \__postnotes_make_text_mark:nnn #1#2#3 ,
217     maketextmark .value_required:n = true ,
218     maketextmark .initial:n = { #2 #1 . #3 } ,
219   }
220 \cs_generate_variant:Nn \__postnotes_make_mark:nnn { Vnn }
```

pretextmark, posttextmark, postprintnote options

```
221 \tl_new:N \l__postnotes_pre_textmark_tl
222 \tl_new:N \l__postnotes_post_textmark_tl
223 \tl_new:N \l__postnotes_post_printnote_tl
224 \keys_define:nn { postnotes/setup }
225   {
226     pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
227     pretextmark .value_required:n = true ,
228     posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
229     posttextmark .value_required:n = true ,
230     postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
231     postprintnote .value_required:n = true ,
232   }
```

hyperref and backlink options

```
233 \bool_new:N \l__postnotes_hyperlink_bool
234 \bool_new:N \l__postnotes_hyperref_warn_bool
235 \bool_new:N \l__postnotes_backlink_bool
236 \keys_define:nn { postnotes/setup }
237   {
238     hyperref .choice: ,
239     hyperref / auto .code:n =
240       {
241         \bool_set_true:N \l__postnotes_hyperlink_bool
242         \bool_set_false:N \l__postnotes_hyperref_warn_bool
243       } ,
244     hyperref / true .code:n =
245       {
246         \bool_set_true:N \l__postnotes_hyperlink_bool
247         \bool_set_true:N \l__postnotes_hyperref_warn_bool
248       } ,
249     hyperref / false .code:n =
250       {
251         \bool_set_false:N \l__postnotes_hyperlink_bool
252         \bool_set_false:N \l__postnotes_hyperref_warn_bool
```



```

253     } ,
254     hyperref .initial:n = auto ,
255     hyperref .default:n = true ,
256     backlink .bool_set:N = \l__postnotes_backlink_bool ,
257     backlink .initial:n = true ,
258     backlink .default:n = true ,
259 }
260 \AddToHook { begindocument }
261 {
262   \IfPackageLoadedTF { hyperref }
263   { }
264   {
265     \bool_if:NT \l__postnotes_hyperref_warn_bool
266     { \msg_warning:nn { postnotes } { missing-hyperref } }
267     \bool_set_false:N \l__postnotes_hyperlink_bool
268   }
269   \keys_define:nn { postnotes/setup }
270   {
271     hyperref .code:n =
272     {
273       \msg_warning:nnn { postnotes }
274       { option-preamble-only } { hyperref }
275     } ,
276     backlink .code:n =
277     {
278       \msg_warning:nnn { postnotes }
279       { option-preamble-only } { backlink }
280     } ,
281   }
282 }
283 \msg_new:nnn { postnotes } { option-preamble-only }
284 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
285 \msg_new:nnn { postnotes } { missing-hyperref }
286 { Missing~'hyperref'~package.~Setting~'hyperref=false'. }

```

multiple, multisep options

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

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

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

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

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

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

```

287 \cs_new_eq:NN \_postnotes_multiple_prepare: \prg_do_nothing:
288 \cs_new_eq:NN \_postnotes_multiple_check: \prg_do_nothing:
289 \cs_new_eq:NN \_postnotes_multiple_store_lastkern: \prg_do_nothing:
290 \tl_new:N \l__postnotes_multisep_tl
291 \tl_const:Nn \c_postnotes_multi_notemarker_tl { 3sp }
292 \dim_new:N \l__postnotes_multi_lastkern_dim
293 \bool_new:N \l__postnotes_multi_stored_lastkern_bool
294 \tl_new:N \l__postnotes_saved_spacefactor_multi_tl
295 \keys_define:nn { postnotes/setup }
296 {
297     multiple .choice: ,
298     multiple / true .code:n =
299     {

```

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`.

```

300     \cs_set_protected:Npn \_postnotes_multiple_prepare:
301     {

```

```

302     \kern -\c_postnotes_multi_notemarker_tl
303     \kern \c_postnotes_multi_notemarker_tl
304     \scan_stop:
305   }
306   \cs_set_protected:Npn \__postnotes_multiple_check:
307   {
308     \dim_compare:nNnT
309     { \c_postnotes_multi_notemarker_tl } =
310     {
311       \bool_if:NTF \l__postnotes_multi_stored_lastkern_bool
312       { \l__postnotes_multi_lastkern_dim }
313       { \lastkern }
314     }
315     {
316       \tl_set:Nc \l__postnotes_saved_spacefactor_multi_tl
317       { \int_use:N \spacefactor }

```

latex-lab-footmisc.ltx comments at this point “shouldn’t that be 2 unkeres?? (none would also be ok)” and goes on to add a second `\unkern`. If these calls are supposed to undo the `\keres` in `__postnotes_multiple_prepare:`, in our case we probably lost those at this point, for the same reason we have to store `\lastkern` at the beginning of `\posnote` (see below). So, we might as well go with “none”, since otherwise we might be removing from the list a kern we didn’t intend to.

```

318         % \unkern
319         % \unkern
320         \tag_socket_use:n { postnotes/multsep/begin }
321         \__postnotes_make_mark:Vnn \l__postnotes_multisep_tl { } { }
322         \tag_socket_use:n { postnotes/multsep/end }
323         \spacefactor \l__postnotes_saved_spacefactor_multi_tl
324         \scan_stop:
325     }
326   }

```

`\postnote` does quite a lot of stuff before actually calling `__postnotes_typeset_mark:eV` which results in `\lastkern` set by the previous `__postnotes_multiple_prepare:` being lost (why?), so we need to store it earlier.

```

327     \cs_set_protected:Npn \__postnotes_multiple_store_lastkern:
328     {
329       \dim_set_eq:NN \l__postnotes_multi_lastkern_dim \lastkern
330       \bool_set_true:N \l__postnotes_multi_stored_lastkern_bool
331     }
332   } ,
333   multiple / false .code:n =
334   {
335     \cs_set_eq:NN \__postnotes_multiple_prepare: \prg_do_nothing:
336     \cs_set_eq:NN \__postnotes_multiple_check: \prg_do_nothing:
337     \cs_set_eq:NN \__postnotes_multiple_store_lastkern: \prg_do_nothing:
338   } ,
339   multiple / unknown .code:n =
340   { \msg_error:nnn { postnotes } { boolean-values-only } { multiple } } ,
341   multiple .default:n = true ,
342   multisep .tl_set:N = \l__postnotes_multisep_tl ,
343   multisep .initial:n = {,} ,
344 }

```

```

345 \msg_new:nmmn { postnotes } { boolean-values-only }
346 { Key~'#1'~accepts~boolean-values-only. }
347 { The~key~'#1'~only~accepts~the~values~'true'~and~'false'. }

```

sort option

```

348 \bool_new:N \l__postnotes_sort_bool
349 \keys_define:nn { postnotes/setup }
350 {
351   sort .bool_set:N = \l__postnotes_sort_bool ,
352   sort .initial:n = true ,
353   sort .default:n = true ,
354 }

```

style option

```

355 \keys_define:nn { postnotes/setup }
356 {
357   style .choice: ,
358   style / endnotes .meta:n =
359   {
360     listenv = none ,
361     format =
362     {
363       \footnotesize
364       \setlength { \rightskip } { Opt }
365       \setlength { \leftskip } { Opt }
366       \setlength { \parindent } { 1.8em }
367     } ,
368     pretextmark = { \par } ,

```

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.

```

369     maketextmark =
370     {
371       \hbox_set:Nn \l__postnotes_tmpa_box
372       { \@textsuperscript { \normalfont ##1 } }
373       \skip_horizontal:n { - \box_wd:N \l__postnotes_tmpa_box }
374       ##2 \box_use:N \l__postnotes_tmpa_box ##3
375     } ,
376   } ,
377   style / pagenote .meta:n =
378   {
379     listenv = none ,
380     format = { } ,
381     pretextmark = { \par\noindent } ,
382     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
383     posttextmark = { ~ } ,
384   } ,
385 }

```

\postnotesetup

\postnotesetup Provide \postnotesetup.

```
\postnotesetup{options}
```

```

386 \NewDocumentCommand \postnotesetup { m }
387   { \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.

```
388 \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`.

```
389 \int_new:N \g__postnotes_note_id_int
390 \tl_new:N \l_postnotes_note_id_tl
391 \tl_set:Nn \l_postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
392 \seq_new:N \g__postnotes_queue_seq
```

(End of definition for `\g__postnotes_note_id_int`, `\l_postnotes_note_id_tl`, and `\g__postnotes_queue_seq`. This function is documented on page ??.)

`\postnote` Provide `\postnote`.

```
\postnote [options] {note text}
```

```
393 \NewDocumentCommand \postnote { 0 { } +m }
394 { \__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>}
395 \NewHook { postnotes/note/begin }
396 \cs_new_protected:Npn \__postnotes_note:nn #1#2
397 {
398   \group_begin:
399   \__postnotes_multiple_store_lastkern:
400   \keys_set:nn { postnotes/note } {#1}
401   \__postnotes_inhibit_note:F
402   {
403     \int_gincr:N \g__postnotes_note_id_int
404     \tl_if_empty:NT \l__postnotes_mark_tl
405     {
406       \stepcounter { postnote }
407       \tl_set:Ne \l__postnotes_mark_tl { \thepostnote }
408     }
409     \seq_gput_right:Ne \g__postnotes_queue_seq
410     { \l__postnotes_note_id_tl }
411     \UseHook { postnotes/note/begin }
412     \cs_set:Npn \@currentcounter { postnote }
413     \cs_set:Npe \@currentlabel { \p@postnote \l__postnotes_mark_tl }
414     \MakeLinkTarget* { postnote. \l__postnotes_note_id_tl .mark }
415     \__postnotes_set_mark_page_label:e { \l__postnotes_note_id_tl }
416     \__postnotes_set_user_labels:
417     \__postnotes_store:nn { \l__postnotes_note_id_tl } {#2}
418     \bool_if:NTF \l__postnotes_nomark_bool
419     { \tag_socket_use:n { postnotes/mark/nomark } }
420     {
421       \__postnotes_typeset_mark:eV
422       { \l__postnotes_note_id_tl } \l__postnotes_mark_tl
423     }
424   }
425   \group_end:
426 }

```

(End of definition for __postnotes_note:nn.)

Options for \postnote.

```

427 \tl_new:N \l__postnotes_mark_tl
428 \bool_new:N \l__postnotes_nomark_bool
429 \fp_new:N \l__postnotes_sort_num_fp
430 \tl_new:N \l__postnotes_note_label_tl
431 \bool_new:N \l__postnotes_manual_sortnum_bool
432 \bool_new:N \l__postnotes_maybe_multi_bool
433 \keys_define:nn { postnotes/note }
434 {
435   markstr .tl_set:N = \l__postnotes_mark_tl ,
436   markstr .value_required:n = true ,
437   sortnum .code:n =
438   {
439     \fp_set:Nn \l__postnotes_sort_num_fp {#1}
440     \bool_set_true:N \l__postnotes_manual_sortnum_bool
441   } ,
442   sortnum .value_required:n = true ,
443   mark .meta:n =

```



```

444     {
445       markstr = {#1} ,
446       sortnum = {#1} ,
447     } ,
448     mark .value_required:n = true ,
449     nomark .bool_set:N = \l__postnotes_nomark_bool ,
450     nomark .default:n = true ,
451     label .tl_set:N = \l__postnotes_note_label_tl ,
452     label .value_required:n = true ,
453   }

```

`__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.

```

454 \bool_new:N \l__postnotes_inhibit_note_bool
455 \bool_new:N \l__postnotes_print_plain_mark_bool
456 \bool_new:N \l__postnotes_print_plain_mark_stepcounter_bool
457 \NewHook { postnotes/note/inhibit }
458 \prg_new_protected_conditional:Npnn \__postnotes_inhibit_note: { F }
459 {
460   \bool_set_false:N \l__postnotes_inhibit_note_bool
461   \bool_set_false:N \l__postnotes_print_plain_mark_bool
462   \bool_set_false:N \l__postnotes_print_plain_mark_stepcounter_bool
463   \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.

```

464   \bool_if:NT \l__postnotes_print_plain_mark_bool
465   {
466     \tl_if_empty:NT \l__postnotes_mark_tl
467     {
468       \bool_if:NTF \l__postnotes_print_plain_mark_stepcounter_bool
469       {
470         \stepcounter { postnote }
471         \tl_set:Ne \l__postnotes_mark_tl { \thepostnote }
472       }
473       {
474         \group_begin:
475         \int_incr:N \c@postnote
476         \exp_args:NNNe
477         \group_end:

```

```

478         \tl_set:Nn \l__postnotes_mark_tl { \thepostnote }
479     }
480 }
481 \group_begin:
482 \socket_assign_plug:nn { tagssupport/postnotes/multsep/begin } { noop }
483 \socket_assign_plug:nn { tagssupport/postnotes/multsep/end } { noop }
484 \__postnotes_typeset_mark_wrapper:nnn
485 { \__postnotes_make_mark:Vnn \l__postnotes_mark_tl { } { } }
486 { } { }
487 \group_end:
488 }
489 \bool_if:NTF \l__postnotes_inhibit_note_bool
490 { \prg_return_true: }
491 { \prg_return_false: }
492 }

```

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

`__postnotes_typeset_mark:nn` 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:nn {<note id>} {<mark>}
\__postnotes_typeset_mark_wrapper:nnn
{<mark>} {<begin tagging>} {<end tagging>}
493 \cs_new_protected:Npn \__postnotes_typeset_mark:nn #1#2
494 {
495     \__postnotes_typeset_mark_wrapper:nnn
496     {
497         \bool_if:NTF \l__postnotes_hyperlink_bool
498         {
499             \__postnotes_make_mark:nnn {#2}
500             { \hyper@linkstart { link } { postnote. #1 .text } }
501             { \hyper@linkend }
502         }
503         { \__postnotes_make_mark:nnn {#2} { } { } }
504     }
505     { \tag_socket_use:n { postnotes/mark/begin } }
506     { \tag_socket_use:n { postnotes/mark/end } }
507 }
508 \cs_generate_variant:Nn \__postnotes_typeset_mark:nn { eV }
509 \tl_new:N \l__postnotes_saved_spacefactor_tl
510 \cs_new_protected:Npn \__postnotes_typeset_mark_wrapper:nnn #1#2#3
511 {
512     \mode_leave_vertical:
513     \mode_if_horizontal:T
514     {
515         \tl_set:Ne \l__postnotes_saved_spacefactor_tl
516         { \int_use:N \spacefactor }
517         \__postnotes_multiple_check:
518         \nobreak
519     }
520     #2 % begin tagging function
521     #1 % mark
522     #3 % end tagging function

```

```

523   \_postnotes_multiple_prepare:
524   \mode_if_horizontal:T
525     { \spacefactor \l__postnotes_saved_spacefactor_tl }
526   \scan_stop:
527   }

```

(End of definition for _postnotes_typeset_mark:nn 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.

```

528 \cs_new_protected:Npn \_postnotes_set_user_labels:
529 {
530   \tl_if_empty:NF \l__postnotes_note_label_tl
531     { \exp_args:NV \label \l__postnotes_note_label_tl }
532   \tl_if_empty:NF \l__postnotes_note_zlabel_tl
533     { \exp_args:NV \zlabel \l__postnotes_note_zlabel_tl }
534 }

```

(End of definition for _postnotes_set_user_labels:.)

5 \postnoteref

\postnoteref Provide \postnoteref.

```

\postnoteref(*){\label}
535 \NewDocumentCommand \postnoteref { s m }
536 { \_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}
537 \tl_new:N \l__postnotes_note_ref_label_tl
538 \cs_new_protected:Npn \_postnotes_note_ref:nn #1#2
539 {
540   \group_begin:
541   \tl_set:Nn \l__postnotes_note_ref_label_tl {#2}
542   \_postnotes_typeset_mark_wrapper:nnn
543     {
544       \bool_lazy_and:nnTF
545         { ! #1 }
546         { \l__postnotes_hyperlink_bool }
547         {
548           \hyperref [#2]
549             { \_postnotes_make_mark:nnn { \ref*{#2} } { } { } }
550         }
551         { \_postnotes_make_mark:nnn { \ref*{#2} } { } { } }
552     }
553   { \tag_socket_use:n { postnotes/postnoteref/begin } }
554   { \tag_socket_use:n { postnotes/postnoteref/end } }
555   \group_end:
556 }

```

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

6 `\postnotesection`

`\postnotesection` Provide `\postnotesection` and `\postnotesectionx`.
`\postnotesectionx`

```
\postnotesection[<options>]{<section content>}
\postnotesectionx[<options>]{<section content>}

557 \NewDocumentCommand \postnotesection { 0 { } +m }
558 { \_postnotes_section:nn {#1} {#2} }
559 \NewDocumentCommand \postnotesectionx { 0 { } +m }
560 {
561   % NOTE Command deprecated in 2022-12-27 for v0.2.0.
562   \msg_warning:nn { postnotes } { postnotesectionx-deprecated }
563   \postnotesection [ #1 , exp ] {#2}
564 }
565 \msg_new:nnn { postnotes } { postnotesectionx-deprecated }
566 {
567   '\iow_char:N\postnotesectionx'~is-deprecated~\msg_line_context:~
568   Use~the~'exp'~option~of~'\iow_char:N\postnotesection'~instead.
569 }
```

(End of definition for `\postnotesection` and `\postnotesectionx`.)

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

```
\_postnotes_section:nn {<options>} {<content>}

570 \int_new:N \g__postnotes_sectid_int
571 \cs_new_protected:Npn \_postnotes_section:nn #1#2
572 {
573   \group_begin:
574   \int_gincr:N \g__postnotes_sectid_int
575   \int_gincr:N \g__postnotes_note_id_int
576   \seq_gput_right:Ne \g__postnotes_queue_seq { \l_postnotes_note_id_tl }
577   \tl_gclear:N \g__postnotes_section_name_tl
578   \keys_set:nn { postnotes/section } {#1}
579   \bool_if:NTF \l__postnotes_section_exp_bool
580     { \_postnotes_store_section:ne { \l_postnotes_note_id_tl } {#2} }
581     { \_postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
582   \group_end:
583 }
```

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

Options for `\postnotesection`. 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.

```
584 \tl_new:N \g__postnotes_section_name_tl
585 \bool_new:N \l__postnotes_section_exp_bool
586 \keys_define:nn { postnotes/section }
587 {
```

```

588     name .tl_gset:N = \g__postnotes_section_name_tl ,
589     name .value_required:n = true ,
590     exp .bool_set:N = \l__postnotes_section_exp_bool ,
591     exp .initial:n = false ,
592     exp .default:n = true ,
593 }

```

7 \printpostnotes

`\printpostnotes` Provide `\printpostnotes`.

```

\printpostnotes
594 \NewDocumentCommand \printpostnotes { }
595 { \__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>	596 <code>\tl_new:N \pnthechapter</code>
<code>\pnthesectionnextnote</code>	597 <code>\tl_new:N \pnthesection</code>
<code>\pnthepage</code>	598 <code>\tl_new:N \pnidnextnote</code>
<code>\pnidnextnote</code>	599 <code>\tl_new:N \pnthechapternextnote</code>
	600 <code>\tl_new:N \pnthesectionnextnote</code>
	601 <code>\tl_new:N \pnthepage</code>

(End of definition for \pnthechapter and others.)

<code>\g__postnotes_print_postnotes_int</code>	Auxiliary variables for <code>__postnotes_print_notes:</code> .
<code>\l__postnotes_print_note_id_tl</code>	602 <code>\int_new:N \g__postnotes_print_postnotes_int</code>
<code>\l__postnotes_print_note_id_next_tl</code>	603 <code>\tl_new:N \l__postnotes_print_note_id_tl</code>
<code>\l__postnotes_print_counter_tl</code>	604 <code>\tl_new:N \l__postnotes_print_note_id_next_tl</code>
<code>\l__postnotes_print_mark_tl</code>	605 <code>\tl_new:N \l__postnotes_print_counter_tl</code>
<code>\l__postnotes_print_type_curr_tl</code>	606 <code>\tl_new:N \l__postnotes_print_mark_tl</code>
<code>\l__postnotes_print_type_next_tl</code>	607 <code>\tl_new:N \l__postnotes_print_type_curr_tl</code>
<code>\l__postnotes_print_type_prev_tl</code>	608 <code>\tl_new:N \l__postnotes_print_type_next_tl</code>
<code>\l__postnotes_print_content_tl</code>	609 <code>\tl_new:N \l__postnotes_print_type_prev_tl</code>
<code>\l__postnotes_clear_queue_seq</code>	610 <code>\tl_new:N \l__postnotes_print_content_tl</code>
	611 <code>\seq_new:N \l__postnotes_clear_queue_seq</code>

(End of definition for \g__postnotes_print_postnotes_int and others. This function is documented on page ??.)

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

```

612 \NewHook { postnotes/print/begin }
613 \NewHook { postnotes/print/note/begin }

```

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

```

614 \newcounter { postnotetext }
615 \newcounter { postnotesection }
616 \setcounter { postnotesection } { 10000 }

```

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

```

\__postnotes_print_notes:
617 \cs_new_protected:Npn \__postnotes_print_notes:
618 {
619   \group_begin:
620   \int_gincr:N \g__postnotes_print_postnotes_int
621   \seq_if_empty:NTF \g__postnotes_queue_seq
622     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
623   {
624     \pnheading
625     \UseHook { postnotes/print/begin }
626     \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
627     \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_queue_seq
628     \__postnotes_verify_multipass:N \g__postnotes_queue_seq
629     \bool_if:NT \l__postnotes_sort_bool
630       { \__postnotes_sort_queue:N \g__postnotes_queue_seq }
631     \bool_gset_true:N \g__postnotes_header_vars_next_bool
632     \__postnotes_get_headers_data:N \g__postnotes_queue_seq
633     \__postnotes_set_headers_vars_first:

```

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

```

634     \bool_if:NF \l__postnotes_print_as_list_bool
635     {

```

```

636         \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
637         \@afterindenttrue
638     }
639 \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_queue_seq }
640 {
641     \seq_gpop_left:NN \g__postnotes_queue_seq
642     \l_postnotes_print_note_id_tl
643     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
644     { type } \l_postnotes_print_type_curr_tl
645     \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
646     { % type_curr = 'section'
647         \seq_if_empty:NnTF \g__postnotes_queue_seq
648         {
649             \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
650             \tl_set:Nn \l__postnotes_print_type_next_tl { close }
651         }
652         {
653             \seq_get_left:NN \g__postnotes_queue_seq
654             \l__postnotes_print_note_id_next_tl
655             \__postnotes_prop_get:nnN
656             { \l__postnotes_print_note_id_next_tl }
657             { type } \l__postnotes_print_type_next_tl
658         }

```

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

```

659     \tl_if_eq:NnTF \l__postnotes_print_type_next_tl { note }
660     {
661         \stepcounter { postnotessection }
662         \group_begin:
663         \__postnotes_prop_get:nnN
664         { \l_postnotes_print_note_id_tl }
665         { thechapter } \pnthechapter
666         \__postnotes_prop_get:nnN
667         { \l_postnotes_print_note_id_tl }
668         { thesection } \pnthesection
669         \tl_set:NV \pnidnextnote \l__postnotes_print_note_id_next_tl
670         \__postnotes_prop_get:nnN
671         { \l__postnotes_print_note_id_next_tl }
672         { thechapter } \pnthechapternextnote
673         \__postnotes_prop_get:nnN
674         { \l__postnotes_print_note_id_next_tl }
675         { thesection } \pnthesectionnextnote
676         \__postnotes_prop_get:nnN
677         { \l_postnotes_print_note_id_tl }
678         { content } \l__postnotes_print_content_tl
679         \l__postnotes_print_content_tl
680         \group_end:

```

Set type_prev for the next iteration.

```

681         \tl_set:NV \l__postnotes_print_type_prev_tl
682         \l__postnotes_print_type_curr_tl
683     }
684 }
685 { % type_curr = 'note'
686     \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }

```

```

687     {
688         \bool_if:NTF \l__postnotes_print_as_list_bool
689         { \exp_args:Ne \begin { \l__postnotes_print_env_tl } }
690         { \group_begin: }
691         \tag_socket_use:n { postnotes/printlist/begin }
692         \l__postnotes_print_format_tl
693     }
694     \group_begin:
695     \UseHook { postnotes/print/note/begin }
696     \__postnotes_get_pageref:Ne \pnthepage
697     { mark@ \l__postnotes_print_note_id_tl }
698     \__postnotes_prop_get:nnN
699     { \l__postnotes_print_note_id_tl }
700     { mark } \l__postnotes_print_mark_tl
701     \__postnotes_prop_get:nnN
702     { \l__postnotes_print_note_id_tl }
703     { counter } \l__postnotes_print_counter_tl
704     \__postnotes_prop_get:nnN
705     { \l__postnotes_print_note_id_tl }
706     { content } \l__postnotes_print_content_tl
707     \cs_set:Npn \@currentcounter { postnotetext }
708     \int_set:Nn \c@postnotetext
709     { \l__postnotes_print_counter_tl }
710     \cs_set:Npe \@currentlabel
711     { \p@postnote \l__postnotes_print_mark_tl }
712     \tag_socket_use:n { postnotes/printnote/begin }
713     \__postnotes_text_mark_wrapper:n
714     {
715         \MakeLinkTarget*
716         { postnote. \l__postnotes_print_note_id_tl .text }
717         \__postnotes_set_text_page_label:e
718         { \l__postnotes_print_note_id_tl }
719         \__postnotes_typeset_text_mark:eV
720         { \l__postnotes_print_note_id_tl }
721         \l__postnotes_print_mark_tl
722     }
723     \tag_socket_use:n { postnotes/printtext/begin }
724     \l__postnotes_print_content_tl
725     \tag_socket_use:n { postnotes/printtext/end }
726     \tag_socket_use:n { postnotes/printnote/end }
727     \l__postnotes_post_printnote_tl
728     \group_end:

```

For notes, query for next note’s type after the current note was typeset, to handle possible nesting. Even if nesting is not a feature, this should avoid hard crashes related to “lonely \item” or “extra \endgroup” errors, in case it occurs.

```

729     \seq_if_empty:NTF \g__postnotes_queue_seq
730     {
731         \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
732         \tl_set:Nn \l__postnotes_print_type_next_tl { close }
733     }
734     {
735         \seq_get_left:NN \g__postnotes_queue_seq
736         \l__postnotes_print_note_id_next_tl

```



```

737         \_postnotes_prop_get:nnN
738         { \l__postnotes_print_note_id_next_tl }
739         { type } \l__postnotes_print_type_next_tl
740     }
741     \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
742     {
743         \tag_socket_use:n { postnotes/printlist/end }
744         \bool_if:NTF \l__postnotes_print_as_list_bool
745         { \exp_args:Ne \end { \l__postnotes_print_env_tl } }
746         { \group_end: }

```

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

```

747         \par
748     }

```

Set `type_prev` for the next iteration.

```

749         \tl_set:NV \l__postnotes_print_type_prev_tl
750         \l__postnotes_print_type_curr_tl
751     }
752 }
753 \AddToHookNext { shipout/after }
754 { \bool_gset_false:N \g__postnotes_header_vars_next_bool }

```

We won't use the variables anymore, clear them to reduce memory usage. Given how we populated `\l__postnotes_clear_queue_seq`, this won't catch nested notes. But it's not worth to conditionally add new items along the way (testing it every iteration) for this. Again, not a feature.

```

755     \seq_map_inline:Nn \l__postnotes_clear_queue_seq
756     { \_postnotes_prop_gclear:n { ##1 } }
757 }
758 \group_end:
759 }

```

(End of definition for `_postnotes_print_notes:`)

```

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

```

`_postnotes_typeset_text_mark:nn`
`_postnotes_text_mark_wrapper:n`

```

Auxiliary functions for mark typesetting in \_postnotes_print_notes:.

    \_postnotes_typeset_text_mark:nn {<note id>} {<mark>}
    \_postnotes_text_mark_wrapper:n {<mark>}

762 \cs_new_protected:Npn \_postnotes_typeset_text_mark:nn #1#2
763 {
764     \bool_lazy_and:nnTF
765     { \l__postnotes_hyperlink_bool }
766     { \l__postnotes_backlink_bool }
767     {
768         \_postnotes_make_text_mark:nnn {#2}
769         { \hyper@linkstart { link } { postnote. #1 .mark } }
770         { \hyper@linkend }
771     }
772     { \_postnotes_make_text_mark:nnn {#2} { } { } }
773 }

```

```

774 \cs_generate_variant:Nn \_postnotes_typeset_text_mark:nn { eV }
775 \cs_new_protected:Npn \_postnotes_text_mark_wrapper:n #1
776 {
777   \bool_if:NTF \l__postnotes_print_as_list_bool
778   {
779     \item
780     [
781       \tag_socket_use:n { postnotes/printmark/begin }
782       \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl
783       \tag_socket_use:n { postnotes/printmark/end }
784     ]

```

Leave vertical mode to avoid “perhaps a missing \item” error for empty notes.

```

785   \mode_leave_vertical:
786 }
787 {
788   \tag_socket_use:n { postnotes/printmark/begin }
789   \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl
790   \tag_socket_use:n { postnotes/printmark/end }
791 }
792 }

```

(End of definition for _postnotes_typeset_text_mark:nn and _postnotes_text_mark_wrapper:n.)

Print auxiliary

_postnotes_verify_multipass: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 \postnotes of measuring/trial passes don’t end up being output and hence don’t generate labels in the .aux file. This is the equivalent for postnotes to the effect of write restrictions for the packages based on external files, which is how they actually handle these cases. However, despite this being a general test, and a reasonable one, I’d like to restrain it’s use to the minimum possible. First, using this criterion across the board would result in large swings on the content of \printpostnotes and spurious warnings in an initial compilation since the labels are not available on the first run. Second, I’d prefer not to interfere with the queue, unless we really need to. 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_verify_multipass:N   \_postnotes_verify_multipass:N (\g__postnotes_queue_seq)
793 \cs_new_protected:Npn \_postnotes_verify_multipass:N #1
794 {
795   \group_begin:
796   \seq_clear:N \l__postnotes_tmpa_seq
797   \seq_map_inline:Nn #1
798   {
799     \_postnotes_prop_get:nnN {##1} { multibool } \l__postnotes_tmpa_tl

```

```

800     \str_if_eq:VnTF \l__postnotes_tmpa_tl { true }
801     {
802         \cs_if_exist:cT
803         { \c__postnotes_ref_prefix_tl @ mark@ ##1 }
804         { \seq_put_right:Nn \l__postnotes_tmpa_seq {##1} }
805     }
806     { \seq_put_right:Nn \l__postnotes_tmpa_seq {##1} }
807 }
808 \seq_gset_eq:NN #1 \l__postnotes_tmpa_seq
809 \group_end:
810 }

```

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

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

```

      \__postnotes_sort_queue:N (\g__postnotes_queue_seq)
811 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
812 {
813     \group_begin:
814     \seq_gsort:Nn #1
815     {
816         \__postnotes_prop_get:nnN {##1} { pnsectid } \l__postnotes_tmpa_tl
817         \__postnotes_prop_get:nnN {##2} { pnsectid } \l__postnotes_tmpb_tl
818         \tl_if_eq:NNTF \l__postnotes_tmpa_tl \l__postnotes_tmpb_tl
819         {
820             \__postnotes_prop_get:nnN {##1} { type } \l__postnotes_tmpa_tl
821             \__postnotes_prop_get:nnN {##2} { type } \l__postnotes_tmpb_tl
822             \bool_lazy_and:nnTF
823             { \str_if_eq_p:Vn \l__postnotes_tmpa_tl { note } }
824             { \str_if_eq_p:Vn \l__postnotes_tmpb_tl { note } }
825             {
826                 \__postnotes_prop_get:nnN {##1} { sortnum } \l__postnotes_tmpa_tl
827                 \__postnotes_prop_get:nnN {##2} { sortnum } \l__postnotes_tmpb_tl
828                 \fp_compare:nNnTF
829                 { \l__postnotes_tmpa_tl } > { \l__postnotes_tmpb_tl }
830                 { \sort_return_swapped: }
831                 { \sort_return_same: }
832             }
833             { \sort_return_same: }
834         }
835         { \sort_return_same: }
836     }
837     \group_end:
838 }

```

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

8 Headers

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

1. For each `\postnote`, labels are set storing the `page` where the note occurs. Each note actually generates a pair of such labels, once when `\postnote` is called (with the mark), and another where the note is printed (in `\printpostnotes`). The former ones store `\thepage`, since we want the printed representation of it for typesetting purposes, the latter ones store the value of the `page` counter, since we don't need to typeset it, but do need to perform algebraic operations with it. These labels are set by `__postnotes_set_mark_page_label:n`, `__postnotes_set_text_page_label:n`, and `__postnotes_set_print_page_label:n` at the appropriate places. The set of these labels provides a mapping from each note's "mark" and "text" to the page where it occurs.
2. This information set is processed by `__postnotes_get_headers_data:N` at the beginning of `__postnotes_print_notes:` to identify the first and last note of each page in `\printpostnotes`, and to generate a mapping from these first and last notes on each page to the pages where their corresponding marks occur. We also take the opportunity to enrich this mapping with other metadata of each note. So we get also mappings from the first and last note on each page to `\thechapter`, `\thesection`, and the `name` of the section in which they occur. These mappings are stored in property lists `\g__postnotes_header_{info}_first_prop` and `\g__postnotes_header_{info}_last_prop` where the key is the page in `\printpostnotes` where their note's content is typeset (or rather where it starts to be typeset, it is the page where the text's mark is printed).
3. Based on these mappings, along the span of notes section we run `__postnotes_set_headers_vars_next:` at each `shipout/before` hook to set user facing variables for the *next* page, which will be available when their heading gets typeset. Given that at `shipout` we can rely on a correct value of the `page` counter, we use it as `key` to query the property lists generated in the previous step. These user facing variables are called `\pnhd{info}first` and `\pnhd{info}last`. Since we cannot rely on the shipout hook for the first page of `\printpostnotes`, `__postnotes_set_headers_vars_first:` is run at its beginning to ensure correct values are in place on the first page of the notes section.

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

<code>\pnhdpagefirst</code>	User facing variables, aimed at making available header data for the user. Setting these
<code>\pnhdpagelast</code>	variables with correct values at the moment the header gets typeset is <i>the</i> objective of
<code>\pnhdchapfirst</code>	the whole headers infrastructure of the package.
<code>\pnhdchaplast</code>	
<code>\pnhdsectfirst</code>	<small>839</small> <code>\tl_new:N \pnhdpagefirst</code>
<code>\pnhdsectlast</code>	<small>840</small> <code>\tl_new:N \pnhdpagelast</code>
<code>\pnhdnamefirst</code>	<small>841</small> <code>\tl_new:N \pnhdchapfirst</code>
<code>\pnhdnamelast</code>	<small>842</small> <code>\tl_new:N \pnhdchaplast</code>
	<small>843</small> <code>\tl_new:N \pnhdsectfirst</code>
	<small>844</small> <code>\tl_new:N \pnhdsectlast</code>
	<small>845</small> <code>\tl_new:N \pnhdnamefirst</code>
	<small>846</small> <code>\tl_new:N \pnhdnamelast</code>

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

Auxiliary variables for the headers infrastructure.

```

\g__postnotes_header_page_first_prop
\g__postnotes_header_page_last_prop
\g__postnotes_header_chap_first_prop
\g__postnotes_header_chap_last_prop
\g__postnotes_header_sect_first_prop
\g__postnotes_header_sect_last_prop
\g__postnotes_header_name_first_prop
\g__postnotes_header_name_last_prop
\g__postnotes_header_prev_last_page_tl
\g__postnotes_header_prev_last_chap_tl
\g__postnotes_header_prev_last_sect_tl
\g__postnotes_header_prev_last_name_tl
  \l__postnotes_prev_text_page_tl
  \l__postnotes_curr_text_page_tl
  \l__postnotes_prev_mark_page_tl
  \l__postnotes_prev_mark_chap_tl
  \l__postnotes_prev_mark_sect_tl
  \l__postnotes_prev_mark_name_tl

```

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

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

```

  \__postnotes_get_headers_data:N <\g__postnotes_queue_seq>
865 \cs_new_protected:Npn \__postnotes_get_headers_data:N #1
866 {
867   \group_begin:
868   \tl_gclear:N \pnhdpagefirst
869   \tl_gclear:N \pnhdpagelast
870   \tl_gclear:N \pnhdchapfirst
871   \tl_gclear:N \pnhdchaplast
872   \tl_gclear:N \pnhdsectfirst
873   \tl_gclear:N \pnhdsectlast
874   \tl_gclear:N \pnhdnamefirst
875   \tl_gclear:N \pnhdnamelast
876   \prop_gclear:N \g__postnotes_header_page_first_prop
877   \prop_gclear:N \g__postnotes_header_page_last_prop
878   \prop_gclear:N \g__postnotes_header_chap_first_prop
879   \prop_gclear:N \g__postnotes_header_chap_last_prop
880   \prop_gclear:N \g__postnotes_header_sect_first_prop
881   \prop_gclear:N \g__postnotes_header_sect_last_prop
882   \prop_gclear:N \g__postnotes_header_name_first_prop
883   \prop_gclear:N \g__postnotes_header_name_last_prop
884   \tl_gclear:N \g__postnotes_header_prev_last_page_tl
885   \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
886   \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
887   \tl_gclear:N \g__postnotes_header_prev_last_name_tl
888   \tl_clear:N \l__postnotes_prev_text_page_tl
889   \tl_clear:N \l__postnotes_curr_text_page_tl
890   \tl_clear:N \l__postnotes_prev_mark_page_tl
891   \tl_clear:N \l__postnotes_prev_mark_chap_tl
892   \tl_clear:N \l__postnotes_prev_mark_sect_tl

```

```

893 \tl_clear:N \l__postnotes_prev_mark_name_tl
894 \seq_map_inline:Nn #1
895 {
896   \exp_args:Ne \tl_if_eq:nnT
897   { \l__postnotes_prop_item:nn {##1} { type } }
898   { note }
899   {
900     \l__postnotes_get_pageref:Nn
901     \l__postnotes_curr_text_page_tl { text@ ##1 }
902     \tl_if_empty:NF \l__postnotes_curr_text_page_tl
903     {
904       \tl_if_eq:NNTF
905       \l__postnotes_prev_text_page_tl
906       \l__postnotes_curr_text_page_tl
907       {

```

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

```

908     \l__postnotes_get_pageref:Nn
909     \l__postnotes_prev_mark_page_tl { mark@ ##1 }
910     \l__postnotes_prop_get:nnN {##1} { thechapter }
911     \l__postnotes_prev_mark_chap_tl
912     \l__postnotes_prop_get:nnN {##1} { thesection }
913     \l__postnotes_prev_mark_sect_tl
914     \l__postnotes_prop_get:nnN {##1} { pnsectname }
915     \l__postnotes_prev_mark_name_tl
916   }
917 {

```

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

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

```

918     \tl_if_empty:NF \l__postnotes_prev_text_page_tl
919     {
920       \prop_gput:Nee \g__postnotes_header_page_last_prop
921       { \l__postnotes_prev_text_page_tl }
922       { \l__postnotes_prev_mark_page_tl }
923       \prop_gput:Nee \g__postnotes_header_chap_last_prop
924       { \l__postnotes_prev_text_page_tl }
925       { \l__postnotes_prev_mark_chap_tl }
926       \prop_gput:Nee \g__postnotes_header_sect_last_prop
927       { \l__postnotes_prev_text_page_tl }
928       { \l__postnotes_prev_mark_sect_tl }
929       \prop_gput:Nee \g__postnotes_header_name_last_prop
930       { \l__postnotes_prev_text_page_tl }
931       { \l__postnotes_prev_mark_name_tl }
932     }

```

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

```

933     \prop_gput:Nee \g__postnotes_header_page_first_prop
934     { \l__postnotes_curr_text_page_tl }
935     { \l__postnotes_extract_pageref:n { mark@ ##1 } }

```

```

936         \prop_gput:Nee \g__postnotes_header_chap_first_prop
937         { \l__postnotes_curr_text_page_tl }
938         { \__postnotes_prop_item:nn {##1} { thechapter } } }
939     \prop_gput:Nee \g__postnotes_header_sect_first_prop
940     { \l__postnotes_curr_text_page_tl }
941     { \__postnotes_prop_item:nn {##1} { thesection } } }
942     \prop_gput:Nee \g__postnotes_header_name_first_prop
943     { \l__postnotes_curr_text_page_tl }
944     { \__postnotes_prop_item:nn {##1} { pnsectname } } }

```

Store `prev_mark` data for the first note on the page.

```

945     \__postnotes_get_pageref:Nn
946     \l__postnotes_prev_mark_page_tl { mark@ ##1 }
947     \__postnotes_prop_get:nnN {##1} { thechapter }
948     \l__postnotes_prev_mark_chap_tl
949     \__postnotes_prop_get:nnN {##1} { thesection }
950     \l__postnotes_prev_mark_sect_tl
951     \__postnotes_prop_get:nnN {##1} { pnsectname }
952     \l__postnotes_prev_mark_name_tl

```

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

```

953         \tl_set:NV \l__postnotes_prev_text_page_tl
954         \l__postnotes_curr_text_page_tl
955     }
956 }
957 }
958 }

```

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

```

959     \tl_if_empty:NF \l__postnotes_prev_text_page_tl
960     {
961         \prop_gput:Nee \g__postnotes_header_page_last_prop
962         { \l__postnotes_prev_text_page_tl }
963         { \l__postnotes_prev_mark_page_tl } }
964     \prop_gput:Nee \g__postnotes_header_chap_last_prop
965     { \l__postnotes_prev_text_page_tl }
966     { \l__postnotes_prev_mark_chap_tl } }
967     \prop_gput:Nee \g__postnotes_header_sect_last_prop
968     { \l__postnotes_prev_text_page_tl }
969     { \l__postnotes_prev_mark_sect_tl } }
970     \prop_gput:Nee \g__postnotes_header_name_last_prop
971     { \l__postnotes_prev_text_page_tl }
972     { \l__postnotes_prev_mark_name_tl } }
973 }
974 \group_end:
975 }

```

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

The sequence of pages processed in `__postnotes_get_headers_data:N` is not ensured to be continuous, since not every page of `\printpostnotes` starts a note. There may be notes that fill whole pages, or the last page of the notes may end with a note

that started on the penultimate page. We must handle this case at `__postnotes_set_headers_vars:n`. For every page for which there is information provided by `__postnotes_get_headers_data:N` we store a `header_prev_last` (the last value of the previous header) for each of the variables of interest. If the next page is skipped in the sequence (no notes starting on it), we can use these stored values to set both ‘first’ and ‘last’ variables based on them for that page.

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

```

\__postnotes_set_headers_vars:n {(page number)}
976 \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
977 {
978   \group_begin:
979   \prop_get:NnNTF \g__postnotes_header_page_first_prop
980     {#1} \l__postnotes_tmpa_tl
981     { \tl_gset:NV \pnhdpagefirst \l__postnotes_tmpa_tl }
982     { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
983   \prop_get:NnNTF \g__postnotes_header_page_last_prop
984     {#1} \l__postnotes_tmpa_tl
985     {
986       \tl_gset:NV \pnhdpagelast \l__postnotes_tmpa_tl
987       \tl_gset:NV \g__postnotes_header_prev_last_page_tl
988         \l__postnotes_tmpa_tl
989     }
990     { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
991   \prop_get:NnNTF \g__postnotes_header_chap_first_prop
992     {#1} \l__postnotes_tmpa_tl
993     { \tl_gset:NV \pnhdchapfirst \l__postnotes_tmpa_tl }
994     { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
995   \prop_get:NnNTF \g__postnotes_header_chap_last_prop
996     {#1} \l__postnotes_tmpa_tl
997     {
998       \tl_gset:NV \pnhdchaplast \l__postnotes_tmpa_tl
999       \tl_gset:NV \g__postnotes_header_prev_last_chap_tl
1000         \l__postnotes_tmpa_tl
1001     }
1002     { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
1003   \prop_get:NnNTF \g__postnotes_header_sect_first_prop
1004     {#1} \l__postnotes_tmpa_tl
1005     { \tl_gset:NV \pnhdsectfirst \l__postnotes_tmpa_tl }
1006     { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
1007   \prop_get:NnNTF \g__postnotes_header_sect_last_prop
1008     {#1} \l__postnotes_tmpa_tl
1009     {
1010       \tl_gset:NV \pnhdsectlast \l__postnotes_tmpa_tl
1011       \tl_gset:NV \g__postnotes_header_prev_last_sect_tl
1012         \l__postnotes_tmpa_tl
1013     }
1014     { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
1015   \prop_get:NnNTF \g__postnotes_header_name_first_prop
1016     {#1} \l__postnotes_tmpa_tl
1017     { \tl_gset:NV \pnhdnamefirst \l__postnotes_tmpa_tl }

```



```

1018     { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
1019     \prop_get:NnNTF \g__postnotes_header_name_last_prop
1020     {#1} \l__postnotes_tmpa_tl
1021     {
1022       \tl_gset:NV \pnhdnamelast \l__postnotes_tmpa_tl
1023       \tl_gset:NV \g__postnotes_header_prev_last_name_tl
1024       \l__postnotes_tmpa_tl
1025     }
1026     { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
1027   \group_end:
1028 }
1029 \cs_generate_variant:Nn \__postnotes_set_headers_vars:n { e }

```

(End of definition for __postnotes_set_headers_vars:n.)

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

```

1030 \AddToHook { shipout/before } [ postnotes/header ]
1031 { \__postnotes_set_headers_vars_next: }
1032 \bool_new:N \g__postnotes_header_vars_next_bool
1033 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:
1034 {
1035   \bool_if:NT \g__postnotes_header_vars_next_bool
1036   { \__postnotes_set_headers_vars:e { \int_eval:n { \c@page + 1 } } }
1037 }
1038 \cs_new_protected:Npn \__postnotes_set_headers_vars_first:
1039 {
1040   \__postnotes_set_print_page_label:e
1041   { \int_use:N \g__postnotes_print_postnotes_int }
1042   \__postnotes_set_headers_vars:e
1043   {
1044     \__postnotes_extract_pageref:e
1045     { print@ \int_use:N \g__postnotes_print_postnotes_int }
1046   }
1047 }

```

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

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

```

\pnheaderdefault
1048 \NewDocumentCommand \pnheaderdefault {}
1049 {
1050   \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
1051   { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
1052   { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
1053 }

```

(End of definition for \pnheaderdefault.)

9 Compatibility

A dedicated temp variable for restoring data.

```
1054 \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.

From the user's perspective, one-line captions will just work. For two-line captions, there are two alternatives: i) decrement the counter by 1 `\addtocounter{postnote}{-1}` before the caption, then call `\postnote` inside the caption; or ii) right before the caption, call `\postnote[nomark]{\label{mynote}...}`, then use `\postnoteref{mynote}` inside the caption.

```
1055 \AddToHook { postnotes/note/begin } [ postnotes ]
1056 {
1057   \cs_if_exist:NT \@captype
1058   { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1059 }
```

`biblatex`

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

```
1060 \AddToHook { package/biblatex/after }
1061 {
```

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

```
1062   \AddToHook { postnotes/print/begin } [ postnotes ]
1063   { \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.

```
1064   \AddToHook { begindocument/before } [ postnotes/mkbibendnote ]
1065   {
1066     \cs_set:Npn \blx@theendnote { \postnote }
1067     \cs_set:Npn \blx@theendnotetext
1068     { \blx@err@endnote \footnotetext }
1069   }
1070 }
1071 <*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):

```
1072 \AddToHook { package/biblatex/after }
1073 {
```

Store biblatex variables for each note.

```
1074   \AddToHook { postnotes/note/store } [ postnotes ]
1075   {
1076     \prop_gput:cne { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1077     { biblatex@refsection } { \int_use:N \c@refsection }
1078     \prop_gput:cne { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1079     { biblatex@refsegment } { \int_use:N \c@refsegment }
1080     \prop_gput:cne { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1081     { biblatex@refcontextbool }
1082     { \iftoggle { blx@refcontext } { true } { false } }
1083     \prop_gput:cnV { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1084     { biblatex@refcontext } \blx@refcontext@context
1085   }
```

biblatex setup, once for \printpostnotes call.

```
1086   \AddToHook { postnotes/print/begin } [ postnotes ]
1087   {
1088     \l__postnotes_biblatex_endrefcontext_local:
1089     \l__postnotes_biblatex_citereset_local:
1090   }
```

Restore biblatex variables for each note.

```
1091   \AddToHook { postnotes/print/note/begin } [ postnotes ]
1092   {
1093     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1094     { biblatex@refsection } \l__postnotes_restore_tmp_tl
1095     \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
1096     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1097     { biblatex@refsegment } \l__postnotes_restore_tmp_tl
1098     \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
1099     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1100     { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
1101     \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
1102     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1103     { biblatex@refcontext } \l__postnotes_restore_tmp_tl
1104     \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
1105   }
```

Auxiliary functions.

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

```
1106   \cs_new_protected:Npn \l__postnotes_biblatex_endrefcontext_local:
1107   {
```

```

1108     \togglefalse { blx@refcontext }
1109     \tl_clear:N \blx@refcontext@labelprefix
1110     \tl_clear:N \blx@refcontext@labelprefix@real
1111     \tl_set:Ne \blx@refcontext@sortingtemplatename { \blx@sorting }
1112     \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
1113     \tl_set:Nn \blx@refcontext@uniquenametemplatenamename { global }
1114     \tl_set:Nn \blx@refcontext@labelalphanametemplatenamename { global }
1115     \blx@edef@refcontext
1116     {
1117         \blx@refcontext@sortingtemplatename /
1118         \blx@refcontext@sortingnamekeytemplatename /
1119         /
1120         \blx@refcontext@uniquenametemplatenamename /
1121         \blx@refcontext@labelalphanametemplatenamename
1122     }
1123 }

```

(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.

```

1124     \cs_new_protected:Npn \__postnotes_biblatex_citereset_local:
1125     {
\global\cslet{blx@bsee@the\c@refsection}\empty
\global\cslet{blx@fsee@the\c@refsection}\empty
1126         \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1127         \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
\blx@ibidreset@force
1128         \undef \blx@lastkey@text
1129         \undef \blx@lastkey@foot
\blx@idemreset@force
1130         \undef \blx@lasthash@text
1131         \undef \blx@lasthash@foot
\blx@opcitreset@force
1132         \clist_map_inline:Nn \blx@trackhash@text
1133         { \csundef { blx@lastkey@text@ ##1 } }
1134         \tl_clear:N \blx@trackhash@text
1135         \clist_map_inline:Nn \blx@trackhash@foot
1136         { \csundef { blx@lastkey@foot@ ##1 } }
1137         \tl_clear:N \blx@trackhash@foot
\blx@loccitreset@force
1138         \clist_map_inline:Nn \blx@trackkeys@text
1139         { \csundef { blx@lastnote@text@ ##1 } }
1140         \tl_clear:N \blx@trackkeys@text
1141         \clist_map_inline:Nn \blx@trackkeys@foot
1142         { \csundef { blx@lastnote@foot@ ##1 } }
1143         \tl_clear:N \blx@trackkeys@foot
and all of them do:
1144         \cs_set_eq:NN \blx@lastmpfn \z@
1145     }

```

(End of definition for __postnotes_biblatex_citereset_local:.)

1146 }

`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`.

```
1147 \AddToHook { package/biblatex/after }
1148 {
1149   \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1150   \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1151   \AddToHook { postnotes/print/begin } [ postnotes ]
1152   {
1153     \tl_set:Nc \l__postnotes_biblatex_orig_refsection_tl
1154       { \int_use:N \c@refsection }
1155     \tl_gset:Nc \g__postnotes_biblatex_prev_refsection_tl
1156       { \l__postnotes_biblatex_orig_refsection_tl }
1157   }
1158   \AddToHook { postnotes/print/note/begin } [ postnotes ]
1159   {
1160     \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1161       { biblatex@refsection } \l__postnotes_restore_tmp_tl
1162     \tl_if_eq:NNF
1163       \l__postnotes_restore_tmp_tl
1164       \g__postnotes_biblatex_prev_refsection_tl
1165       {
1166         \int_set:Nn \c@blx@maxsection
1167           { \l__postnotes_restore_tmp_tl - 1 }
1168         \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1169           \l__postnotes_restore_tmp_tl
1170         \group_begin:
1171         \cs_set_eq:NN \label \use_none:n
1172         \cs_set_eq:NN \blx@info \use_none:n
1173         \blx@endrefsection
1174         \refsection
1175         \group_end:
1176       }
1177   }
1178 }
1179 </gobble>
```

zref-user

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

```
1180 \tl_new:N \l__postnotes_note_zlabel_tl
```

(End of definition for `\l__postnotes_note_zlabel_tl`.)

```
1181 \AddToHook { package/zref-user/after }
1182   {
```

Provide `zlabel` option.

```
1183   \keys_define:nn { postnotes/note }
1184   {
1185     zlabel .tl_set:N = \l__postnotes_note_zlabel_tl ,
1186     zlabel .value_required:n = true ,
1187   }
```

`\postnotezref` Provide `\postnotezref`.

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

```
1188 \NewDocumentCommand \postnotezref { s m }
1189   { \__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)}
```

```
1190 \tl_new:N \l__postnotes_note_zref_zlabel_tl
1191 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1192   {
1193     \group_begin:
1194     \tl_set:Nn \l__postnotes_note_zref_zlabel_tl {#2}
1195     \__postnotes_typeset_mark_wrapper:nnn
1196     {
1197       \bool_lazy_all:nTF
1198       {
1199         { ! #1 }
1200         { \l__postnotes_hyperlink_bool }
1201         { \l__postnotes_zrefhyperref_bool }
1202       }
1203       {
1204         \hyperlink
1205         { \zref@extractdefault {#2} { anchor } { } }
1206         { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1207       }
1208       { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1209     }
1210     { \tag_socket_use:n { postnotes/postnotezref/begin } }
1211     { \tag_socket_use:n { postnotes/postnotezref/end } }
1212   \group_end:
1213 }
```

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

```
1214 }
1215 \bool_new:N \l__postnotes_zrefhyperref_bool
1216 \AddToHook { package/zref-hyperref/after }
1217 { \bool_set_true:N \l__postnotes_zrefhyperref_bool }
```

zref-clever

```
1218 \AddToHook { package/zref-clever/after }
1219 {
1220   \zcsetup
1221   {
1222     countertype = { postnote = endnote } ,
1223     countertype = { postnotetext = endnote } ,
1224   }
1225   \AddToHook { postnotes/print/begin } [ postnotes ]
1226   { \zcsetup { counterresetby = { postnotetext = postnotesection } } }
1227 }
```

zref-check

```
1228 \AddToHook { package/zref-check/after }
1229 {
1230   \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1231   {
1232     \AddToHook { postnotes/note/store } [ postnotes ]
1233     {
1234       \prop_gput:cne { \_postnotes_data_name:e { \l__postnotes_note_id_tl } }
1235       { zref-check@abschap } { \int_use:N \c@zc@abschap }
1236       \prop_gput:cne { \_postnotes_data_name:e { \l__postnotes_note_id_tl } }
1237       { zref-check@abssec } { \int_use:N \c@zc@abssec }
1238     }
1239     \AddToHook { postnotes/print/note/begin } [ postnotes ]
1240     {
1241       \_postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1242       { zref-check@abschap } \l__postnotes_restore_tmp_tl
1243       \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1244       \_postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1245       { zref-check@abssec } \l__postnotes_restore_tmp_tl
1246       \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1247     }
1248   }
1249   { }
1250 }
```

amsmath

```
1251 \AddToHook { package/amsmath/after }
1252 {
```

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

```
1253   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1254   {
```

```

1255     \legacy_if:nT { measuring@ }
1256     {
1257         \bool_set_true:N \l__postnotes_inhibit_note_bool
1258         \bool_set_true:N \l__postnotes_print_plain_mark_bool
1259         \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1260     }
1261 }

```

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 `\l__postnotes_maybe_multi_bool` 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.)”

```

1262     \AddToHook { postnotes/note/begin } [ postnotes ]
1263     {
1264         \legacy_if:nF { firstchoice@ }
1265         { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1266     }
1267 }

```

csquotes

```

1268 \AddToHook { package/csquotes/after }
1269 {
1270     \bool_new:N \l__postnotes_csquotes_measuring_bool
1271     \BlockquoteDisable
1272     { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1273     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1274     {
1275         \bool_if:NT \l__postnotes_csquotes_measuring_bool
1276         {
1277             \bool_set_true:N \l__postnotes_inhibit_note_bool
1278             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1279             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1280         }
1281     }
1282 }

```

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>.

```

1283 \AddToHook { package/tabularx/after }
1284 {
1285     \bool_new:N \l__postnotes_tabularx_inside_env_bool
1286     \AddToHook { env/tabularx/begin } [ postnotes ]

```



```

1287     {
1288     \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1289     \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1290     }
1291 \AddToHook { postnotes/note/inhibit } [ postnotes ]
1292 {
1293     \bool_lazy_and:nnT
1294     { \l__postnotes_tabularx_inside_env_bool }
1295     { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1296     {
1297         \bool_set_true:N \l__postnotes_inhibit_note_bool
1298         \bool_set_true:N \l__postnotes_print_plain_mark_bool
1299         \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1300     }
1301 }
1302 }

```

tabularray

```

1303 \AddToHook { package/tabularray/after }
1304 {

```

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).

```

1305     \bool_if_exist:NTF \lTblrMeasuringBool
1306     {

```

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, `pkgpostnotes` expects and requires the counter module.

```

1307     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1308     {
1309         \bool_if:NT \lTblrMeasuringBool
1310         {
1311             \bool_set_true:N \l__postnotes_inhibit_note_bool
1312             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1313             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1314         }
1315     }
1316 }
1317 {

```

If the new boolean is not yet available, we use `__postnotes_verify_multipass:N` to distinguish a trial/measure pass from the final one.

```

1318     \clist_map_inline:nn
1319     {
1320         tblr , longtblr , talltblr , booktabs ,

```

```

1321         longtabs , talltabs , +array
1322     }
1323     {
1324         \AddToHook { env/#1/begin } [ postnotes ]
1325         { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1326     }
1327 }
1328 }

```

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.

```

1329 \socket_new:nn { tagssupport/postnotes/mark/begin }{ 0 }
1330 \socket_new:nn { tagssupport/postnotes/mark/end }{ 0 }
1331 \socket_new:nn { tagssupport/postnotes/mark/nomark }{ 0 }
1332 \socket_new:nn { tagssupport/postnotes/multsep/begin }{ 0 }
1333 \socket_new:nn { tagssupport/postnotes/multsep/end }{ 0 }
1334 \socket_new:nn { tagssupport/postnotes/printlist/begin }{ 0 }
1335 \socket_new:nn { tagssupport/postnotes/printlist/end }{ 0 }
1336 \socket_new:nn { tagssupport/postnotes/printnote/begin }{ 0 }
1337 \socket_new:nn { tagssupport/postnotes/printnote/end }{ 0 }
1338 \socket_new:nn { tagssupport/postnotes/printmark/begin }{ 0 }
1339 \socket_new:nn { tagssupport/postnotes/printmark/end }{ 0 }
1340 \socket_new:nn { tagssupport/postnotes/printtext/begin }{ 0 }
1341 \socket_new:nn { tagssupport/postnotes/printtext/end }{ 0 }
1342 \socket_new:nn { tagssupport/postnotes/postnoteref/begin }{ 0 }
1343 \socket_new:nn { tagssupport/postnotes/postnoteref/end }{ 0 }
1344 \socket_new:nn { tagssupport/postnotes/postnotezref/begin }{ 0 }
1345 \socket_new:nn { tagssupport/postnotes/postnotezref/end }{ 0 }

1346 \bool_lazy_and:nnT
1347   { \cs_if_exist_p:N \tag_if_active_p: }
1348   { \tag_if_active_p: }
1349   {

```

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

```

1350   \tagpdfsetup
1351   {
1352     role/new-tag = { tag=endnote, role=FENote } ,
1353     role/new-tag = { tag=endnotemark, role=Lbl } ,
1354     role/new-tag = { tag=endnotelabel, role=Lbl } ,
1355     role/new-attribute =
1356       { EndnoteType } { /O /FENote /NoteType /Endnote } ,
1357     role/new-attribute =
1358       { EndnotesList } { /O /List /ListNumbering /Ordered } ,
1359   }

```

`\postnote`

```

1360   \socket_new_plug:nnn { tagssupport/postnotes/mark/begin } { default }
1361   {
1362     \tag_mc_end_push:
1363     \tag_struct_begin:n
1364     {
1365       tag = endnotemark ,
1366       label = { postnotemark. \l_postnotes_note_id_tl } ,

```

```

1367         ref = { postnote. \l_postnotes_note_id_tl } ,
1368     }
1369     \__postnotes_tagsup_store_sstructnum:nN
1370     { postnotemark } \l_postnotes_note_id_tl
1371     \tag_mc_begin:n{
1372 }
1373 \socket_new_plug:nnn { tagsupport/postnotes/mark/end } { default }
1374 {
1375     \tag_mc_end:
1376     \tag_struct_end: % endnotemark
1377     \tag_mc_begin_pop:n{
1378 }
1379 \socket_new_plug:nnn { tagsupport/postnotes/mark/nomark } { default }
1380 {
1381     \tag_struct_begin:n
1382     {
1383         tag = NonStruct ,
1384         label = { postnotemark. \l_postnotes_note_id_tl } ,
1385         ref = { postnote. \l_postnotes_note_id_tl } ,
1386     }
1387     \__postnotes_tagsup_store_sstructnum:nN
1388     { postnotemark } \l_postnotes_note_id_tl
1389     \tag_struct_end: % NonStruct
1390 }
1391 \socket_assign_plug:nn { tagsupport/postnotes/mark/begin } { default }
1392 \socket_assign_plug:nn { tagsupport/postnotes/mark/end } { default }
1393 \socket_assign_plug:nn { tagsupport/postnotes/mark/nomark } { default }

```

multiple

```

1394 \socket_new_plug:nnn { tagsupport/postnotes/multsep/begin } { default }
1395 {
1396     \tag_mc_end_push:
1397     \tag_mc_begin:n { artifact }
1398 }
1399 \socket_new_plug:nnn { tagsupport/postnotes/multsep/end } { default }
1400 {
1401     \tag_mc_end:
1402     \tag_mc_begin_pop:n{
1403 }
1404 \socket_assign_plug:nn { tagsupport/postnotes/multsep/begin } { default }
1405 \socket_assign_plug:nn { tagsupport/postnotes/multsep/end } { default }

```

\printpostnotes

```

1406 \socket_new_plug:nnn { tagsupport/postnotes/printlist/begin } { default }
1407 { \tag_tool:n { para/tagging=false } }
1408 \socket_new_plug:nnn { tagsupport/postnotes/printlist/end } { default }
1409 { }
1410 \socket_assign_plug:nn { tagsupport/postnotes/printlist/begin } { default }
1411 \socket_assign_plug:nn { tagsupport/postnotes/printlist/end } { default }
1412 \socket_new_plug:nnn { tagsupport/postnotes/printnote/begin } { default }
1413 {
1414     \bool_if:NF \l__postnotes_print_as_list_bool
1415     {
1416         \tag_struct_begin:n
1417         {

```

```

1418         tag = endnote ,
1419         attribute-class = EndnoteType ,
1420         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.
1421         ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1422     }
1423     \__postnotes_tagsup_store_sstructnum:nN
1424     { postnote } \l_postnotes_print_note_id_tl
1425 }
1426 }
1427 \socket_new_plug:nnn { tagsupport/postnotes/printnote/end } { default }
1428 {
1429     \bool_if:NF \l__postnotes_print_as_list_bool
1430     { \tag_struct_end: } % endnote
1431 }
1432 \socket_assign_plug:nn { tagsupport/postnotes/printnote/begin } { default }
1433 \socket_assign_plug:nn { tagsupport/postnotes/printnote/end } { default }
1434 \socket_new_plug:nnn { tagsupport/postnotes/printmark/begin } { default }
1435 {
1436     \bool_if:NF \l__postnotes_print_as_list_bool
1437     {
1438         \tag_struct_begin:n { tag=endnotelabel }
1439         \tag_mc_begin:n{ tag=Lbl }
1440     }
1441 }
1442 \socket_new_plug:nnn { tagsupport/postnotes/printmark/end } { default }
1443 {
1444     \bool_if:NF \l__postnotes_print_as_list_bool
1445     {
1446         \tag_mc_end:
1447         \tag_struct_end: % endnotelabel
1448     }
1449 }
1450 \socket_assign_plug:nn { tagsupport/postnotes/printmark/begin } { default }
1451 \socket_assign_plug:nn { tagsupport/postnotes/printmark/end } { default }
1452 \socket_new_plug:nnn { tagsupport/postnotes/printtext/begin } { default }
1453 {
1454     \bool_if:NTF \l__postnotes_print_as_list_bool
1455     {
1456         \tag_struct_begin:n
1457         {
1458             tag = endnote ,
1459             attribute-class = EndnoteType ,
1460             label = { postnote. \l_postnotes_print_note_id_tl } ,

```

CHECK Ditto.

```

1461         ref = { postnotemark. \l_postnotes_print_note_id_tl } ,
1462     }
1463     \__postnotes_tagsup_store_sstructnum:nN
1464     { postnote } \l_postnotes_print_note_id_tl
1465     \tag_struct_begin:n { tag=text-unit }
1466     \tag_struct_begin:n { tag=text }
1467     \tag_tool:n { para/tagging=true }

```

```

1468         \tag_mc_begin:n{
1469     }
1470     {
1471         \tag_struct_begin:n { tag=text-unit }
1472         \tag_struct_begin:n { tag=text }
1473         \tag_tool:n { para/tagging=true }
1474         \tag_mc_begin:n{
1475     }
1476 }
1477 \socket_new_plug:nmn { tagsupport/postnotes/printtext/end } { default }
1478 {
1479     \bool_if:NTF \l__postnotes_print_as_list_bool
1480     {
1481         \tag_mc_end:
1482         \tag_tool:n { para/tagging=false }
1483         \tag_struct_end: % text
1484         \tag_struct_end: % text-unit
1485         \tag_struct_end: % endnote
1486     }
1487     {
1488         \tag_mc_end:
1489         \tag_tool:n { para/tagging=false }
1490         \tag_struct_end: % text
1491         \tag_struct_end: % text-unit
1492     }
1493 }
1494 \socket_assign_plug:mn { tagsupport/postnotes/printtext/begin } { default }
1495 \socket_assign_plug:mn { tagsupport/postnotes/printtext/end } { default }

```

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

```

1496 \IfInstanceExistsT { blockenv } { list }
1497 {
1498     \DeclareInstance { blockenv } { postnoteslist } { display }
1499     {
1500         env-name      = postnoteslist ,
1501         tag-name      = L ,
1502         tag-class     = EndnotesList ,
1503         tagging-recipe = list ,
1504         inner-level-counter = ,
1505         level-increase = true ,
1506         setup-code    = ,
1507         block-instance = list ,
1508         inner-instance = postnoteslist ,
1509     }
1510     \DeclareInstanceCopy { blockenv }
1511     { postnoteslisthang } { postnoteslist }
1512     \EditInstance { blockenv } { postnoteslisthang }
1513     { env-name = postnoteslisthang }
1514     \DeclareInstance { list } { postnoteslist } { std }
1515     { item-instance = postnoteslist }
1516     \DeclareInstance { item } { postnoteslist } { std }
1517     {

```

```

1518         label-format = { \hspace { \labelsep } \normalfont ~ #1 } ,
1519         label-align = left ,
1520     }
1521 \RenewDocumentEnvironment { postnoteslist } { }
1522 {
1523     \UseInstance { blockenv } { postnoteslist }
1524     {
1525         leftmargin      = Opt ,
1526         label-width     = Opt ,
1527         item-indent     = .5\parindent ,
1528         rightmargin    = Opt ,
1529         parindent      = \parindent ,
1530         par-skip       = \parskip ,
1531         item-skip      = Opt ,
1532         beginsep       = .5\topsep ,
1533         begin-par-skip = .5\partopsep ,
1534     }
1535 }
1536 { \endblockenv }
1537 \RenewDocumentEnvironment { postnoteslisthang } { }
1538 {
1539     \UseInstance { blockenv } { postnoteslisthang }
1540     {
1541         leftmargin      = 1em ,
1542         label-width     = -\leftmargin ,
1543         item-indent     = -2\leftmargin ,
1544         rightmargin    = Opt ,
1545         parindent      = \parindent ,
1546         par-skip       = \parskip ,
1547         item-skip      = Opt ,
1548         beginsep       = .5\topsep ,
1549         begin-par-skip = .5\partopsep ,
1550     }
1551 }
1552 { \endblockenv }
1553 }

```

Setup for \label and \zlabel inside the note.

```

1554 \bool_new:N \l__postnotes_inside_note_bool
1555 \AddToHookWithArguments { label } [ postnotes/tagsup ]
1556 {
1557     \bool_if:NT \l__postnotes_inside_note_bool
1558     {
1559         \property_record:nn { postnote@label@innote. #1 }
1560         { postnotes/tagsup@noteid }
1561     }
1562 }
1563 \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1564 { \bool_set_true:N \l__postnotes_inside_note_bool }
1565 \property_new:nnnn { postnotes/tagsup@noteid } { now } { 0 }
1566 { \l__postnotes_print_note_id_tl }
1567 \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
1568 {
1569     \zref@newprop { postnotes@tagsup@noteid } [ 0 ]
1570     { \l__postnotes_print_note_id_tl }

```



```

1571     \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1572     { \zref@localaddprop { main } { postnotes@tagsup@noteid } }
1573   }

```

Setup for label and zlabel options.

```

1574   \AddToHook { postnotes/note/store } [ postnotes/tagsup ]
1575   {
1576     \tl_if_empty:NF \l__postnotes_note_label_tl
1577     {
1578       \prop_gput:cnV
1579       { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1580       { label } \l__postnotes_note_label_tl
1581     }
1582   }
1583   \AddToHook { package/zref-user/after } [ postnotes/tagsup ]
1584   {
1585     \AddToHook { postnotes/note/store } [ postnotes/tagsup ]
1586     {
1587       \tl_if_empty:NF \l__postnotes_note_zlabel_tl
1588       {
1589         \prop_gput:cnV
1590         { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1591         { zlabel } \l__postnotes_note_zlabel_tl
1592       }
1593     }
1594   }
1595   \AddToHook { postnotes/print/note/begin } [ postnotes/tagsup ]
1596   {
1597     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1598     { label } \l__postnotes_restore_tmp_tl
1599     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1600     {
1601       \exp_args:Ne \property_record:nn
1602       { postnote@label@option. \l__postnotes_restore_tmp_tl }
1603       { postnotes/tagsup@noteid }
1604     }
1605     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1606     { zlabel } \l__postnotes_restore_tmp_tl
1607     \tl_if_empty:NF \l__postnotes_restore_tmp_tl
1608     {
1609       \exp_args:Ne \property_record:nn
1610       { postnote@zlabel@option. \l__postnotes_restore_tmp_tl }
1611       { postnotes/tagsup@noteid }
1612     }
1613   }

```

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

```

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

```

```

1614 \prop_new:N \g__postnotes_tagsup_structnums_prop
1615 \cs_new_protected:Npn \__postnotes_tagsup_store_sstructnum:nN #1#2
1616 {
1617   \prop_gput:Nee \g__postnotes_tagsup_structnums_prop
1618   { #1 . #2 } { \tag_get:n { struct_num } }
1619 }
1620 \prop_new:N \g__postnotes_tagsup_crossrefs_prop
1621 \cs_new_protected:Npn \__postnotes_tagsup_store_crossref:nN #1#2
1622 {
1623   \prop_gput:Nee \g__postnotes_tagsup_crossrefs_prop
1624   { \tag_get:n { struct_num } } { #1 . #2 }
1625 }

```

(End of definition for `__postnotes_tagsup_store_sstructnum:nN` `__postnotes_tagsup_store_crossref:nN`.)

```

\__postnotes_tagsup_gput_ref:nn
\__postnotes_tagsup_gput_ref:nn {(structnum referring from)}
  {(structnum being referenced to)}
See \__fnote_gput_ref:nn.
1626 \cs_new_protected:Npn \__postnotes_tagsup_gput_ref:nn #1#2
1627 {
1628   \tag_if_active:T
1629   { \tag_struct_gput:ene {#1} {ref} { \tag_struct_object_ref:e {#2} } }
1630 }

```

(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 `\printposnotes`, and also so that the “main” reference always comes first at the list.

```

1631 \AddToHook { tagpdf/finish/before } [ postnotes/tagsup ]
1632 {
1633   \prop_map_inline:Nn \g__postnotes_tagsup_crossrefs_prop
1634   {
1635     \__postnotes_tagsup_gput_ref:nn
1636     { \prop_item:Nn \g__postnotes_tagsup_structnums_prop {#2} }
1637     {#1}
1638   }
1639 }

```

`\postnoteref`

```

1640 \socket_new_plug:nnn { tagssupport/postnotes/postnoteref/begin } { default }
1641 {
1642   \tag_mc_end_push:
1643   \property_if_recorded:eeTF
1644   { postnote@label@innote. \l__postnotes_note_ref_label_tl }
1645   { postnotes/tagsup@noteid }
1646   {

```

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

```

1647   \tl_set:Ne \l__postnotes_tmpa_tl
1648   {
1649     \property_ref:ee
1650     { postnote@label@innote. \l__postnotes_note_ref_label_tl }

```

```

1651         { postnotes/tagstup@noteid }
1652     }
1653     \tag_struct_begin:n
1654     {
1655         tag = endnotemark ,
1656         ref = { postnote. \l__postnotes_tmpa_tl } ,
1657     }
1658     \__postnotes_tagstup_store_crossref:nN
1659     { postnote } \l__postnotes_tmpa_tl
1660 }
1661 {
1662     \property_if_recorded:eeTF
1663     { postnote@label@option. \l__postnotes_note_ref_label_tl }
1664     { postnotes/tagstup@noteid }
1665     {

```

Label coming from a label option.

```

1666         \tl_set:Ne \l__postnotes_tmpa_tl
1667         {
1668             \property_ref:ee
1669             { postnote@label@option. \l__postnotes_note_ref_label_tl }
1670             { postnotes/tagstup@noteid }
1671         }
1672     \tag_struct_begin:n
1673     {
1674         tag = endnotemark ,
1675         ref = { postnotemark. \l__postnotes_tmpa_tl } ,
1676     }
1677     \__postnotes_tagstup_store_crossref:nN
1678     { postnotemark } \l__postnotes_tmpa_tl
1679 }
1680 { \tag_struct_begin:n { tag = endnotemark } }
1681 }
1682 \tag_mc_begin:n{ }
1683 }
1684 \socket_new_plug:nmn { tagssupport/postnotes/postnoteref/end } { default }
1685 {
1686     \tag_mc_end:
1687     \tag_struct_end: % endnotemark
1688     \tag_mc_begin_pop:n{ }
1689 }
1690 \socket_assign_plug:mn { tagssupport/postnotes/postnoteref/begin } { default }
1691 \socket_assign_plug:mn { tagssupport/postnotes/postnoteref/end } { default }

```

`\postnotezref`

```

1692 \AddToHook { package/zref-user/after } [ postnotes/tagstup ]
1693 {
1694     \socket_new_plug:nmn { tagssupport/postnotes/postnotezref/begin } { default }
1695     {
1696         \tag_mc_end_push:
1697         \zref@ifrefcontainsprop { \l__postnotes_note_zref_zlabel_tl }
1698         { postnotes@tagstup@noteid }
1699         {

```

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

```

1700     \tl_set:Ne \l__postnotes_tmpa_tl
1701     {
1702         \zref@extract { \l__postnotes_note_zref_zlabel_tl }
1703         { postnotes@tagsup@noteid }
1704     }
1705     \tag_struct_begin:n
1706     {
1707         tag = endnotemark ,
1708         ref = { postnote. \l__postnotes_tmpa_tl } ,
1709     }
1710     \__postnotes_tagsup_store_crossref:nN
1711     { postnote } \l__postnotes_tmpa_tl
1712 }
1713 {
1714     \property_if_recorded:eeTF
1715     { postnote@zlabel@option. \l__postnotes_note_zref_zlabel_tl }
1716     { postnotes/tagsup@noteid }
1717     {

```

Label coming from a zlabel option.

```

1718     \tl_set:Ne \l__postnotes_tmpa_tl
1719     {
1720         \property_ref:ee
1721         {
1722             postnote@zlabel@option.
1723             \l__postnotes_note_zref_zlabel_tl
1724         }
1725         { postnotes/tagsup@noteid }
1726     }
1727     \tag_struct_begin:n
1728     {
1729         tag = endnotemark ,
1730         ref = { postnotemark. \l__postnotes_tmpa_tl } ,
1731     }
1732     \__postnotes_tagsup_store_crossref:nN
1733     { postnotemark } \l__postnotes_tmpa_tl
1734 }
1735 { \tag_struct_begin:n { tag = endnotemark } }
1736 }
1737 \tag_mc_begin:n{}
1738 }
1739 \socket_new_plug:nnn { tagsupport/postnotes/postnotezref/end } { default }
1740 {
1741     \tag_mc_end:
1742     \tag_struct_end: % endnotemark
1743     \tag_mc_begin_pop:n{}
1744 }
1745 \socket_assign_plug:nn { tagsupport/postnotes/postnotezref/begin } { default }
1746 \socket_assign_plug:nn { tagsupport/postnotes/postnotezref/end } { default }
1747 }
1748 }

```

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` 1749 `\tl_new:N \pntitle`

`\pnhdtopages` 1750 `\tl_new:N \pnhdnotes`

1751 `\tl_new:N \pnhdtopage`

1752 `\tl_new:N \pnhdtopages`

1753 `\tl_set:Nn \pntitle { Notes }`

1754 `\tl_set:Nn \pnhdnotes { Notes }`

1755 `\tl_set:Nn \pnhdtopage { to~page }`

1756 `\tl_set:Nn \pnhdtopages { to~pages }`

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

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

```

\__postnotes_define_language:nn {\postnote language} {\setup}

1757 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
1758 {
1759   \tl_new:c { g__postnotes_language_ #1 _tl }
1760   \tl_gset:cn { g__postnotes_language_ #1 _tl } {#2}
1761 }

```

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

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

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

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

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

```

    \_postnotes_set_babel_language:nn {(babel language)} {(postnote language)}
1762 \cs_new_protected:Npn \_postnotes_set_babel_language:nn #1#2
1763 {
1764   \ActivateGenericHook { babel/#1/beforeextras }
1765   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
1766     { g__postnotes_language_ #2 _tl }
1767 }

```

(End of definition for _postnotes_set_babel_language:nn.)

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

_postnotes_set_polyglossia_language:nn Sets ⟨polyglossia language⟩ to execute the setup defined by _postnotes_define_language:nn for ⟨postnote language⟩ at the polyglossia \captions⟨language⟩ hook.

```

    \_postnotes_set_polyglossia_language:nn {(polyglossia language)}
      {(postnote language)}
1768 \cs_new_protected:Npn \_postnotes_set_polyglossia_language:nn #1#2
1769 {
1770   \AddToHook { package/polyglossia/after }
1771     {
1772       \exp_args:Nnv \csgappto { captions #1 }
1773         { g__postnotes_language_ #2 _tl }
1774     }
1775 }

```

(End of definition for _postnotes_set_polyglossia_language:nn.)

English

```

1776 \_postnotes_define_language:nn { english }
1777 {
1778   \tl_set:Nn \pntitle { Notes }
1779   \tl_set:Nn \pnhdnotes { Notes }
1780   \tl_set:Nn \pnhdtopage { to~page }
1781   \tl_set:Nn \pnhdtopages { to~pages }
1782 }
1783 \_postnotes_set_babel_language:nn { english } { english }
1784 \_postnotes_set_babel_language:nn { british } { english }
1785 \_postnotes_set_babel_language:nn { american } { english }
1786 \_postnotes_set_babel_language:nn { canadian } { english }
1787 \_postnotes_set_babel_language:nn { australian } { english }
1788 \_postnotes_set_babel_language:nn { newzealand } { english }
1789 \_postnotes_set_babel_language:nn { UKenglish } { english }

```

```

1790 \_postnotes_set_babel_language:nn { USenglish } { english }
1791 \_postnotes_set_polyglossia_language:nn { english } { english }

```

Portuguese

```

1792 \_postnotes_define_language:nn { portuguese }
1793 {
1794   \tl_set:Nn \pntitle      { Notas }
1795   \tl_set:Nn \pnhdnotes   { Notas }
1796   \tl_set:Nn \pnhdtopage  { da-página }
1797   \tl_set:Nn \pnhdtopages { das-páginas }
1798 }
1799 \_postnotes_set_babel_language:nn { portuguese } { portuguese }
1800 \_postnotes_set_babel_language:nn { brazilian } { portuguese }
1801 \_postnotes_set_babel_language:nn { portuges } { portuguese }
1802 \_postnotes_set_babel_language:nn { brazil } { portuguese }
1803 \_postnotes_set_polyglossia_language:nn { portuguese } { portuguese }

```

French

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

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

```

1804 \_postnotes_define_language:nn { french }
1805 {
1806   \tl_set:Nn \pntitle      { Notes }
1807   \tl_set:Nn \pnhdnotes   { Notes }
1808   \tl_set:Nn \pnhdtopage  { de-la-page }
1809   \tl_set:Nn \pnhdtopages { des-pages }
1810 }
1811 \_postnotes_set_babel_language:nn { french } { french }
1812 \_postnotes_set_babel_language:nn { acadian } { french }
1813 \_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`.

```

1814 \_postnotes_define_language:nn { german }
1815 {
1816   \tl_set:Nn \pntitle      { Endnoten }
1817   \tl_set:Nn \pnhdnotes   { Endnoten }
1818   \tl_set:Nn \pnhdtopage  { zu-Seite }
1819   \tl_set:Nn \pnhdtopages { zu-Seiten }
1820 }
1821 \_postnotes_set_babel_language:nn { german } { german }
1822 \_postnotes_set_babel_language:nn { ngerman } { german }
1823 \_postnotes_set_babel_language:nn { austrian } { german }
1824 \_postnotes_set_babel_language:nn { naustrian } { german }
1825 \_postnotes_set_babel_language:nn { swissgerman } { german }
1826 \_postnotes_set_babel_language:nn { nswissgerman } { german }
1827 \_postnotes_set_polyglossia_language:nn { german } { german }
1828 </package>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A	
<code>\ActivateGenericHook</code>	1764
<code>\addto</code>	<u>53</u>
<code>\addtocounter</code>	<u>40</u>
<code>\AddToHook</code>	80, 88, 260, 1030, 1055, 1060, 1062, 1064, 1072, 1074, 1086, 1091, 1147, 1151, 1158, 1181, 1216, 1218, 1225, 1228, 1232, 1239, 1251, 1253, 1262, 1268, 1273, 1283, 1286, 1291, 1303, 1307, 1324, 1563, 1567, 1571, 1574, 1583, 1585, 1595, 1631, 1692, 1765, 1770
<code>\AddToHookNext</code>	753
<code>\AddToHookWithArguments</code>	1555
B	
<code>\begin</code>	689
<code>\BlockquoteDisable</code>	1271
bool commands:	
<code>\bool_gset_false:N</code>	754
<code>\bool_gset_true:N</code>	631
<code>\bool_if:NTF</code> 34, 265, 311, 418, 464, 468, 489, 497, 579, 629, 634, 688, 744, 777, 1035, 1275, 1309, 1414, 1429, 1436, 1444, 1454, 1479, 1557	
<code>\bool_if_exist:NTF</code>	1305
<code>\bool_lazy_all:nTF</code>	1197
<code>\bool_lazy_and:nnTF</code>	544, 764, 822, 1293, 1346
<code>\bool_new:N</code> 155, 233, 234, 235, 293, 348, 428, 431, 432, 454, 455, 456, 585, 1032, 1215, 1270, 1285, 1554	
<code>\bool_set_false:N</code>	162, 242, 251, 252, 267, 460, 461, 462
<code>\bool_set_true:N</code>	167, 241, 246, 247, 330, 440, 1058, 1217, 1257, 1258, 1259, 1265, 1272, 1277, 1278, 1279, 1288, 1297, 1298, 1299, 1311, 1312, 1313, 1325, 1564
<code>\bool_to_str:N</code>	50
<code>\bool_until_do:nn</code>	639
box commands:	
<code>\box_new:N</code>	19
<code>\box_use:N</code>	374
<code>\box_wd:N</code>	373
C	
<code>\caption</code>	13, 14, <u>34</u>
<code>\chapter</code>	136
<code>\citereset</code>	36
clist commands:	
<code>\clist_map_inline:Nn</code>	1132, 1135, 1138, 1141
<code>\clist_map_inline:nn</code>	1318
<code>\counterwithin</code>	15
cs commands:	
<code>\cs_generate_variant:Nn</code> 22, 67, 106, 109, 112, 119, 126, 220, 508, 774, 1029	
<code>\cs_if_eq_p:NN</code>	1295
<code>\cs_if_exist:NTF</code>	38, 58, 115, 122, 132, 802, 1057
<code>\cs_if_exist_p:N</code>	1347
<code>\cs_new:Npn</code>	20, 73, 120, 208
<code>\cs_new_eq:NN</code>	287, 288, 289
<code>\cs_new_protected:Npe</code>	78
<code>\cs_new_protected:Npn</code>	24, 54, 68, 75, 96, 104, 107, 110, 113, 134, 141, 396, 493, 510, 528, 538, 571, 617, 762, 775, 793, 811, 865, 976, 1033, 1038, 1106, 1124, 1191, 1615, 1621, 1626, 1757, 1762, 1768
<code>\cs_set:Npe</code>	413, 710
<code>\cs_set:Npn</code>	412, 707, 1066, 1067
<code>\cs_set_eq:NN</code>	181, 198, 335, 336, 337, 636, 1144, 1171, 1172, 1289
<code>\cs_set_protected:Npn</code> .	300, 306, 327
<code>\csgappto</code>	1772
<code>\csundef</code>	1133, 1136, 1139, 1142
D	
<code>\DeclareInstance</code>	1498, 1514, 1516
<code>\DeclareInstanceCopy</code>	1510
<code>\def</code>	3
dim commands:	
<code>\dim_compare:nNnTF</code>	308
<code>\dim_new:N</code>	292
<code>\dim_set_eq:NN</code>	329
<code>\DocumentMetadata</code>	42
E	
<code>\EditInstance</code>	1512
<code>\end</code>	745
<code>\endblockenv</code>	1536, 1552
<code>\endgroup</code>	24
<code>\endlist</code>	190, 207
<code>\endnotemark</code>	13
<code>\endnotetext</code>	13
<code>\endrefcontext</code>	35

<code>\noteheading</code>	22	<code>\int_use:N</code> ...	31, 36, 48, 108, 111, 317, 391, 516, 1041, 1045, 1077, 1079, 1126, 1127, 1154, 1235, 1237
exp commands:		iow commands:	
<code>\exp_args:Ne</code>	689, 745, 896, 1601, 1609	<code>\iow_char:N</code>	567, 568, 761
<code>\exp_args:NNNe</code>	476	<code>\iow_now:Nn</code>	84, 92
<code>\exp_args:Nnv</code>	1765, 1772	<code>\iow_shipout_x:Nn</code>	5, 100
<code>\exp_args:NV</code>	531, 533	<code>\item</code>	24, 26, 43, 779
<code>\exp_not:N</code>	79	<code>\itemindent</code>	180, 197
<code>\exp_not:n</code>	123	<code>\itemsep</code>	185, 202
F			
<code>\fmtversion</code>	5	K	
fnote internal commands:		<code>\kern</code>	11, 302, 303
<code>__fnote_gput_ref:nn</code>	50	keys commands:	
<code>\footnote</code>	9, 40	<code>\keys_define:nn</code>	
<code>\footnotemark</code>	9, 13–15	127, 148, 156, 210, 224, 236, 269, 295, 349, 355, 433, 586, 1183
<code>\footnotesize</code>	363	<code>\keys_set:nn</code>	387, 400, 578
<code>\footnotetext</code>	13, 14, 1068	L	
<code>\footref</code>	49	<code>\label</code>	14, 43, 48, 50, 531, 1171
fp commands:		<code>\labelsep</code>	209, 1518
<code>\fp_compare:nNnTF</code>	828	<code>\labelwidth</code>	179, 196
<code>\fp_new:N</code>	429	<code>\lastkern</code>	11, 313, 329
<code>\fp_set:Nn</code>	439	<code>\leftmargin</code> .	178, 195, 196, 197, 1542, 1543
<code>\fp_use:N</code>	35	<code>\leftskip</code>	365
G			
<code>\gappto</code>	53	legacy commands:	
group commands:		<code>\legacy_if:nTF</code> .	82, 90, 98, 1255, 1264
<code>\group_begin:</code>	398, 474, 481, 540, 573, 619, 662, 690, 694, 795, 813, 867, 978, 1170, 1193	<code>\list</code>	176, 193
<code>\group_end:</code>	425, 477, 487, 555, 582, 680, 728, 746, 758, 809, 837, 974, 1027, 1175, 1212	<code>\listparindent</code>	183, 200
H			
<code>\hbox</code>	215	<code>\lTblrMeasuringBool</code>	41, 1305, 1309
hbox commands:		M	
<code>\hbox_set:Nn</code>	371	<code>\makeatletter</code>	54
<code>\hspace</code>	209, 1518	<code>\makeatother</code>	54
<code>\hyperlink</code>	1204	<code>\makelabel</code>	181, 198
<code>\hyperref</code>	548	<code>\MakeLinkTarget</code>	2, 414, 715
I			
<code>\IfFormatAtLeastTF</code>	5, 6	<code>\mathchoice</code>	13, 40
<code>\IfInstanceExistsT</code>	1496	<code>\mbox</code>	13
<code>\IfPackageAtLeastTF</code>	1230	<code>\mkbibendnote</code>	34
<code>\IfPackageLoadedTF</code>	262	mode commands:	
<code>\iftoggle</code>	1082	<code>\mode_if_horizontal:TF</code>	513, 524
int commands:		<code>\mode_leave_vertical:</code>	512, 785
<code>\int_eval:n</code>	1036	msg commands:	
<code>\int_gincr:N</code>	403, 574, 575, 620	<code>\msg_error:nnn</code>	340
<code>\int_incr:N</code>	475	<code>\msg_line_context:</code>	284, 567, 761
<code>\int_new:N</code>	389, 570, 602	<code>\msg_new:nnn</code>	283, 285, 565, 760
<code>\int_set:Nn</code>		<code>\msg_new:nnnn</code>	345
... 708, 1095, 1098, 1166, 1243, 1246		<code>\msg_warning:nn</code>	266, 562, 622
		<code>\msg_warning:nnn</code>	273, 278
N			
		<code>\multfootsep</code>	10
		<code>\multiplefootnotemarker</code>	9, 10
		<code>\NeedsTeXFormat</code>	4

`\newcounter` 388, 614, 615
`\NewDocumentCommand` 386,
 393, 535, 557, 559, 594, 1048, 1188
`\NewDocumentEnvironment` 174, 191
`\NewHook` 3, 23, 395, 457, 612, 613
`\newlabel` 4
`\NewTblrEnviron` 41
`\nobreak` 518
`\noindent` 381
`\normalfont` 209, 215, 372, 382, 1518

P

`\PackageError` 9
`\par` 25, 163, 368, 381, 747
`\parindent`
 180, 183, 200, 366, 1527, 1529, 1545
`\parsep` 184, 201
`\parskip` 184, 201, 1530, 1546
`\partopsep` 187, 204, 1533, 1549
`\pnhdchapfirst` 839, 870, 993, 994
`\pnhdchaplast` 839, 871, 998, 1002
`\pnhdnamefirst` 839, 874, 1017, 1018
`\pnhdnamelast` 839, 875, 1022, 1026
`\pnhdnotes` 1051,
 1052, 1749, 1779, 1795, 1807, 1817
`\pnhdpagefirst`
 839, 868, 981, 982, 1050, 1051, 1052
`\pnhdpagelast` 839, 869, 986, 990, 1050, 1052
`\pnhdsectfirst` 839, 872, 1005, 1006
`\pnhdsectlast` 839, 873, 1010, 1014
`\pnhdtopage`
 1051, 1749, 1780, 1796, 1808, 1818
`\pnhdtopages`
 1052, 1749, 1781, 1797, 1809, 1819
`\pnheaderdefault` .. 33, 53, 137, 144, 1048
`\pnheading` 6, 129, 132, 624
`\pnidnextnote` 596, 669
`\pnthechapter` 596, 665
`\pnthechapternextnote` 596, 672
`\pnthepage` 596, 696
`\pnthesection` 596, 668
`\pnthesectionnextnote` 596, 675
`\pntitle`
 136, 143, 1749, 1778, 1794, 1806, 1816
`\posnote` 11
`\posnoteref` 50
`\postnote` 2, 3, 11, 13–16,
 19, 26, 28, 34, 40, 41, 43, 44, 393, 1066
`\postnotemark` 14, 15
`\postnoteref` 14, 15, 19, 40, 43, 44, 50, 535
 postnotes commands:
 `\c_postnotes_multi_notemarker_tl`
 291, 302, 303, 309

`\l_postnotes_note_id_tl`
 15, 389, 410, 414, 415,
 417, 422, 576, 580, 581, 1076, 1078,
 1080, 1083, 1234, 1236, 1366, 1367,
 1370, 1384, 1385, 1388, 1579, 1590
`\l_postnotes_print_note_id_tl` ...
 43, 602, 642, 643, 664, 667,
 677, 697, 699, 702, 705, 716, 718,
 720, 1093, 1096, 1099, 1102, 1160,
 1241, 1244, 1420, 1421, 1424, 1460,
 1461, 1464, 1566, 1570, 1597, 1605

postnotes internal commands:

`\l__postnotes_backlink_bool`
 235, 256, 766
`__postnotes_biblatex_citereset_-`
 local: 1089, 1124, 1124
`__postnotes_biblatex_endrefcontext_-`
 local: 1088, 1106, 1106
`\l_postnotes_biblatex_orig_-`
 refsection_tl . 37, 1149, 1153, 1156
`\g_postnotes_biblatex_prev_-`
 refsection_tl 1150, 1155, 1164, 1168
`\l__postnotes_clear_queue_seq` ...
 25, 602, 627, 755
`\l__postnotes_csquotes_measuring_-`
 bool 1270, 1272, 1275
`\l__postnotes_curr_text_page_tl` .
 31, 847, 889,
 901, 902, 906, 934, 937, 940, 943, 954
`__postnotes_data_name:n`
 2, 15, 20, 20, 22, 26, 27, 28,
 30, 32, 40, 43, 45, 47, 49, 51, 56, 57,
 60, 63, 65, 70, 74, 76, 1076, 1078,
 1080, 1083, 1234, 1236, 1579, 1590
`__postnotes_define_language:mn` .
 53,
 54, 1757, 1757, 1776, 1792, 1804, 1814
`__postnotes_extract_pageref:n` ..
 5, 113, 120, 126, 935, 1044
`__postnotes_get_headers_data:N` .
 28, 29, 31, 32, 632, 865, 865
`__postnotes_get_pageref:Nn`
 .. 5, 113, 113, 119, 696, 900, 908, 945
`\g_postnotes_header_chap_first_-`
 prop 847, 878, 936, 991
`\g_postnotes_header_chap_last_-`
 prop 847, 879, 923, 964, 995
`\g_postnotes_header_name_first_-`
 prop 847, 882, 942, 1015
`\g_postnotes_header_name_last_-`
 prop 847, 883, 929, 970, 1019
`\g_postnotes_header_page_first_-`
 prop 847, 876, 933, 979

<code>\g__postnotes_header_page_last_</code>	<code>\l__postnotes_nomark_bool</code>
<code>prop</code> 847, 877, 920, 961, 983 418, 428, 449
<code>\g__postnotes_header_prev_last_</code>	<code>__postnotes_note:nn</code>
<code>chap_tl</code> . . . 847, 885, 994, 999, 1002 16, 18, 19, 394, 395, 396
<code>\g__postnotes_header_prev_last_</code>	<code>\g__postnotes_note_id_int</code>
<code>name_tl</code> . . 847, 887, 1018, 1023, 1026 15, 389, 403, 575
<code>\g__postnotes_header_prev_last_</code>	<code>\l__postnotes_note_label_tl</code>
<code>page_tl</code> 847, 884, 982, 987, 990 430, 451, 530, 531, 1576, 1580
<code>\g__postnotes_header_prev_last_</code>	<code>__postnotes_note_ref:nn</code>
<code>sect_tl</code> . . 847, 886, 1006, 1011, 1014 19, 536, 537, 538
<code>\g__postnotes_header_sect_first_</code>	<code>\l__postnotes_note_ref_label_tl</code>
<code>prop</code> 847, 880, 939, 1003 537, 541, 1644, 1650, 1663, 1669
<code>\g__postnotes_header_sect_last_</code>	<code>\l__postnotes_note_zlabel_tl</code>
<code>prop</code> 847, 881, 926, 967, 1007 38, 532, 533, 1180, 1185, 1587, 1591
<code>\g__postnotes_header_vars_next_</code>	<code>__postnotes_note_zref:nn</code>
<code>bool</code> 33, 631, 754, 1032, 1035 38, 1189, 1190, 1191
<code>\l__postnotes_hyperlink_bool</code>	<code>\l__postnotes_note_zref_zlabel_</code>
. 233, 241, 246, 251, 267, 497, 546, 765, 1200	<code>tl</code> 1190, 1194, 1697, 1702, 1715, 1723
<code>\l__postnotes_hyperref_warn_bool</code>	<code>\l__postnotes_post_printnote_tl</code>
. 234, 242, 247, 252, 265 163, 223, 230, 727
<code>__postnotes_inhibit_note:</code>	<code>\l__postnotes_post_textmark_tl</code>
. 458 222, 228, 782, 789
<code>__postnotes_inhibit_note:TF</code>	<code>\l__postnotes_pre_textmark_tl</code>
. 26, 401, 454 221, 226, 782, 789
<code>\l__postnotes_inhibit_note_bool</code>	<code>\l__postnotes_prev_mark_chap_tl</code>
. 17, 454, 460, 489, 1257, 1277, 1297, 1311 847, 891, 911, 925, 948, 966
<code>\l__postnotes_inside_note_bool</code>	<code>\l__postnotes_prev_mark_name_tl</code>
. 1554, 1557, 1564 847, 893, 915, 931, 952, 972
<code>__postnotes_list_makelabel:n</code>	<code>\l__postnotes_prev_mark_page_tl</code>
. 181, 198, 208 847, 890, 909, 922, 946, 963
<code>__postnotes_make_mark:nnn</code>	<code>\l__postnotes_prev_mark_sect_tl</code>
. 10, 212, 220, 321, 485, 499, 503, 549, 551, 1206, 1208 847, 892, 913, 928, 950, 969
<code>__postnotes_make_text_mark:nnn</code>	<code>\l__postnotes_prev_text_page_tl</code>
. 216, 768, 772 30, 31, 847, 888, 905, 918, 921, 924, 927, 930, 953, 959, 962, 965, 968, 971
<code>\l__postnotes_manual_sortnum_</code>	<code>\l__postnotes_print_as_list_bool</code>
<code>bool</code> 34, 431, 440 155, 162, 167, 634, 688, 744, 777, 1414, 1429, 1436, 1444, 1454, 1479
<code>\l__postnotes_mark_tl</code>	<code>\l__postnotes_print_content_tl</code>
. 29, 404, 407, 413, 422, 427, 435, 466, 471, 478, 485 602, 678, 679, 706, 724
<code>\l__postnotes_maybe_multi_bool</code>	<code>\l__postnotes_print_counter_tl</code>
. 26, 40, 50, 432, 1058, 1265, 1325 602, 703, 709
<code>\l__postnotes_multi_lastkern_dim</code>	<code>\l__postnotes_print_env_tl</code>
. 292, 312, 329 154, 164, 168, 689, 745
<code>\l__postnotes_multi_stored_</code>	<code>\l__postnotes_print_format_tl</code>
<code>lastkern_bool</code> 293, 311, 330 147, 150, 692
<code>__postnotes_multiple_check:</code>	<code>\l__postnotes_print_mark_tl</code>
. 288, 306, 336, 517 602, 700, 711, 721
<code>__postnotes_multiple_prepare:</code>	<code>\l__postnotes_print_note_id_</code>
. 10, 11, 287, 300, 335, 523	<code>next_tl</code> 602, 649, 654, 656, 669, 671, 674, 731, 736, 738
<code>__postnotes_multiple_store_</code>	<code>__postnotes_print_notes:</code>
<code>lastkern:</code> 289, 327, 337, 399 21, 22, 25, 27, 28, 595, 617, 617
<code>\l__postnotes_multisep_tl</code>	
. 290, 321, 342	

\l__postnotes_print_plain_mark_-
 bool 17,
 455, 461, 464, 1258, 1278, 1298, 1312
\l__postnotes_print_plain_mark_-
 stepcounter_bool 17, 41,
 456, 462, 468, 1259, 1279, 1299, 1313
\g__postnotes_print_postnotes_-
 int 602, 620, 1041, 1045
\l__postnotes_print_type_curr_tl
 602, 644, 645, 682, 750
\l__postnotes_print_type_next_tl
 602, 650, 657, 659, 732, 739, 741
\l__postnotes_print_type_prev_tl
 602, 626, 681, 686, 749
__postnotes_prop_gcLEAR:n
 4, 68, 75, 756
__postnotes_prop_get:nnN
 4, 68, 68,
 643, 655, 663, 666, 670, 673, 676,
 698, 701, 704, 737, 799, 816, 817,
 820, 821, 826, 827, 910, 912, 914,
 947, 949, 951, 1093, 1096, 1099,
 1102, 1160, 1241, 1244, 1597, 1605
__postnotes_prop_item:nn
 4, 68, 73, 897, 938, 941, 944
\g__postnotes_queue_seq .. 15, 26,
 27, 29, 389, 409, 576, 621, 627, 628,
 630, 632, 639, 641, 647, 653, 729, 735
\c__postnotes_ref_prefix_tl
 4, 77, 79, 115, 116, 122, 123, 803
\l__postnotes_restore_tmp_tl ...
 ... 1054, 1094, 1095, 1097, 1098,
 1100, 1101, 1103, 1104, 1161, 1163,
 1167, 1169, 1242, 1243, 1245, 1246,
 1598, 1599, 1602, 1606, 1607, 1610
\l__postnotes_saved_spacefactor_-
 multi_tl 294, 316, 323
\l__postnotes_saved_spacefactor_-
 tl 509, 515, 525
\g__postnotes_sectid_int 48, 570, 574
__postnotes_section:nn
 20, 558, 570, 571
\l__postnotes_section_exp_bool ..
 579, 585, 590
\g__postnotes_section_name_tl ...
 46, 577, 584, 588
__postnotes_set_babel_language:nn
 54, 1762, 1762, 1783, 1784,
 1785, 1786, 1787, 1788, 1789, 1790,
 1799, 1800, 1801, 1802, 1811, 1812,
 1821, 1822, 1823, 1824, 1825, 1826
__postnotes_set_headers_vars:n .
 29, 32, 33, 976, 976, 1029, 1036, 1042
__postnotes_set_headers_vars_-
 first: 28, 33, 633, 1030, 1038
__postnotes_set_headers_vars_-
 next: 28, 33, 1030, 1031, 1033
__postnotes_set_label:nn
 5, 96, 96, 105, 108, 111
__postnotes_set_mark_page_-
 label:n 5, 28, 96, 104, 106, 415
__postnotes_set_polyglossia_-
 language:nn 54,
 1768, 1768, 1791, 1803, 1813, 1827
__postnotes_set_print_page_-
 label:n ... 5, 28, 96, 110, 112, 1040
__postnotes_set_text_page_-
 label:n 5, 28, 96, 107, 109, 717
__postnotes_set_user_labels: ...
 38, 416, 528, 528
\l__postnotes_sort_bool 348, 351, 629
\l__postnotes_sort_num_fp 35, 429, 439
__postnotes_sort_queue:N
 27, 630, 811, 811
__postnotes_store:nn . 3, 23, 24, 417
__postnotes_store_section:nn ...
 3, 54, 54, 67, 580, 581
\l__postnotes_tabularx_inside_-
 env_bool 1285, 1288, 1294
__postnotes_tabularx_saved_-
 write:Nn 1289, 1295
\g__postnotes_tagsup_crossrefs_-
 prop 1620, 1623, 1633
__postnotes_tagsup_gput_ref:nn .
 50, 1626, 1626, 1635
__postnotes_tagsup_store_-
 crossref:nN
 49, 1621, 1658, 1677, 1710, 1732
__postnotes_tagsup_store_-
 sstructnum:nN
 49, 1369, 1387, 1423, 1463, 1615
__postnotes_tagsup_store_-
 sstructnum:nN__postnotes_-
 tagsup_store_crossref:nN ... 1614
\g__postnotes_tagsup_structnums_-
 prop 1614, 1617, 1636
__postnotes_text_mark_wrapper:n
 25, 713, 762, 775
\l__postnotes_tmpa_box
 16, 371, 373, 374
\l__postnotes_tmpa_seq
 16, 796, 804, 806, 808
\l__postnotes_tmpa_tl 16, 799, 800,
 816, 818, 820, 823, 826, 829, 980,
 981, 984, 986, 988, 992, 993, 996,
 998, 1000, 1004, 1005, 1008, 1010,
 1012, 1016, 1017, 1020, 1022, 1024,

1647, 1656, 1659, 1666, 1675, 1678, 1700, 1708, 1711, 1718, 1730, 1733	\providecommand 5, 85, 93
\l__postnotes_tmpb_tl	\ProvidesExplPackage 14
. 16, 817, 818, 821, 824, 827, 829	
__postnotes_typeset_mark:nn	R
. 11, 18, 421, 493, 493, 508	\ref 2, 549, 551
__postnotes_typeset_mark_- wrapper:nnn	\refsection 37, 1174
. 18, 484, 493, 495, 510, 542, 1195	\refstepcounter 15
__postnotes_typeset_text_- mark:nn 25, 719, 762, 762, 774	\RenewDocumentEnvironment 1521, 1537
__postnotes_verify_multipass:N	\rightmargin 182, 199
. 26, 41, 628, 793, 793	\rightskip 364
\l__postnotes_zrefhyperref_bool	S
. 1201, 1215, 1217	scan commands:
\postnotesection 3, 20, 557	\scan_stop: 304, 324, 526
\postnotesectionx 20, 557	\section 143
\postnotesetup 12, 386	seq commands:
\postnotetext 14, 15	\seq_clear:N 796
\postnotezref 38, 44, 51, 1188	\seq_get_left:NN 653, 735
\postnoteref 43	\seq_gpop_left:NN 641
prg commands:	\seq_gput_right:Nn 409, 576
\prg_do_nothing:	\seq_gset_eq:NN 808
. 287, 288, 289, 335, 336, 337	\seq_gsort:Nn 814
\prg_new_protected_conditional:Npnn	\seq_if_empty:NTF 621, 647, 729
. 458	\seq_if_empty_p:N 639
\prg_return_false: 491	\seq_map_inline:Nn 755, 797, 894
\prg_return_true: 490	\seq_new:N 18, 392, 611
\printpostnotes 25	\seq_put_right:Nn 804, 806
\printposnotes 50	\seq_set_eq:NN 627
\printposstones 43	\setcounter 40, 616
\printpostnotes 14, 15, 21, 22, 26, 28, 30, 31, 33, 35, 36, 43-45, 594	\setlength 178, 179, 180, 182, 183, 184, 185, 186, 187, 195, 196, 197, 199, 200, 201, 202, 203, 204, 364, 365, 366
prop commands:	\setlocalecaption 53
\prop_gclear:N 76, 876, 877, 878, 879, 880, 881, 882, 883	skip commands:
\prop_get:NnNTF 70, 979, 983, 991, 995, 1003, 1007, 1015, 1019	\skip_horizontal:n 373
\prop_gput:Nnn	\small 151
. 27, 28, 30, 32, 40, 43, 45, 47, 49, 51, 57, 60, 63, 65, 920, 923, 926, 929, 933, 936, 939, 942, 961, 964, 967, 970, 1076, 1078, 1080, 1083, 1234, 1236, 1578, 1589, 1617, 1623	socket commands:
\prop_item:Nn 74, 1636	\socket_assign_plug:nn 482, 483, 1391, 1392, 1393, 1404, 1405, 1410, 1411, 1432, 1433, 1450, 1451, 1494, 1495, 1690, 1691, 1745, 1746
\prop_map_inline:Nn 1633	\socket_new:nn
\prop_new:N 26, 56, 847, 848, 849, 850, 851, 852, 853, 854, 1614, 1620 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345
property commands:	\socket_new_plug:nnn
\property_if_recorded:nnTF 1360, 1373, 1379, 1394, 1399, 1406, 1408, 1412, 1427, 1434, 1442, 1452, 1477, 1640, 1684, 1694, 1739
. 1643, 1662, 1714	sort commands:
\property_new:nnnn 1565	\sort_return_same: 831, 833, 835
\property_record:nn 1559, 1601, 1609	\sort_return_swapped: 830
\property_ref:nn 1649, 1668, 1720	\space 11
	\spacefactor 317, 323, 516, 525

<code>\stepcounter</code>	15, 406, 470, 661	<code>\blx@lastkey@foot</code>	1129
str commands:		<code>\blx@lastkey@text</code>	1128
<code>\str_if_eq:nnTF</code>	800	<code>\blx@lastmpfn</code>	1144
<code>\str_if_eq_p:nn</code>	823, 824	<code>\blx@refcontext@context</code>	1084
		<code>\blx@refcontext@labelalphanameemplatename</code>	1114, 1121
T		<code>\blx@refcontext@labelprefix</code> ..	1109
tag commands:		<code>\blx@refcontext@labelprefix@real</code>	1110
<code>\tag_get:n</code>	1618, 1624	<code>\blx@refcontext@sortingnamekeyemplatename</code>	1112, 1118
<code>\tag_if_active:TF</code>	1628	<code>\blx@refcontext@sortingtemplatenam</code>	1111, 1117
<code>\tag_if_active_p:</code>	1347, 1348	<code>\blx@refcontext@uniquenamemplatename</code>	1113, 1120
<code>\tag_mc_begin:n</code>	1371, 1397, 1439, 1468, 1474, 1682, 1737	<code>\blx@sorting</code>	1111
<code>\tag_mc_begin_pop:n</code>	1377, 1402, 1688, 1743	<code>\blx@theendnote</code>	1066
<code>\tag_mc_end:</code>	1375, 1401, 1446, 1481, 1488, 1686, 1741	<code>\blx@theendnotetext</code>	1067
<code>\tag_mc_end_push:</code>	1362, 1396, 1642, 1696	<code>\blx@trackhash@foot</code>	1135, 1137
<code>\tag_socket_use:n</code> 320, 322, 419, 505, 506, 553, 554, 691, 712, 723, 725, 726, 743, 781, 783, 788, 790, 1210, 1211		<code>\blx@trackhash@text</code>	1132, 1134
<code>\tag_struct_begin:n</code>	50, 1363, 1381, 1416, 1438, 1456, 1465, 1466, 1471, 1472, 1653, 1672, 1680, 1705, 1727, 1735	<code>\blx@trackkeys@foot</code>	1141, 1143
<code>\tag_struct_end:</code>	1376, 1389, 1430, 1447, 1483, 1484, 1485, 1490, 1491, 1687, 1742	<code>\blx@trackkeys@text</code>	1138, 1140
<code>\tag_struct_gput:nnn</code>	1629	<code>\c@blx@maxsection</code>	1166
<code>\tag_struct_object_ref:n</code>	1629	<code>\c@page</code>	108, 111, 1036
<code>\tag_tool:n</code> 1407, 1467, 1473, 1482, 1489		<code>\c@postnote</code>	17, 31, 36, 475
<code>\tagpdfsetup</code>	1350	<code>\c@postnotetext</code>	708
\TeX and $\LaTeX 2_{\epsilon}$ commands:		<code>\c@refsection</code>	1077, 1095, 1126, 1127, 1154
<code>\@afterindentfalse</code>	636	<code>\c@refsegment</code>	1079, 1098
<code>\@afterindenttrue</code>	22, 636, 637	<code>\c@z@abschap</code>	1235, 1243
<code>\@auxout</code>	100	<code>\c@z@abssec</code>	1237, 1246
<code>\@capttype</code>	1057	<code>\FN@m@f@check</code>	10
<code>\@currentcounter</code>	412, 707	<code>\FN@m@f@prepare</code>	9, 10
<code>\@currentlabel</code>	413, 710	<code>\hyper@linkend</code>	501, 770
<code>\@footnotemark</code>	18	<code>\hyper@linkstart</code>	500, 769
<code>\@ifl@t@r</code>	5	<code>\ifmeasuring@</code>	39
<code>\@mainaux</code>	84	<code>\p@postnote</code>	413, 711
<code>\@makefnmark</code>	8	<code>\post@note</code>	4, 77, 85, 93, 101
<code>\@mkboth</code>	137, 144	<code>\postnotes@required@kernel</code> 3, 4, 6, 11	
<code>\@newl@bel</code>	4, 79	<code>\z@</code>	1144
<code>\@partaux</code>	92	<code>\zref@extract</code>	1702
<code>\@textsuperscript</code>	215, 372	<code>\zref@extractdefault</code>	1205
<code>\blx@bibfiles</code>	37	<code>\zref@ifrefcontainsprop</code>	1697
<code>\blx@edef@refcontext</code>	1104, 1115	<code>\zref@localaddprop</code>	1572
<code>\blx@endrefsection</code>	1173	<code>\zref@newprop</code>	1569
<code>\blx@err@endnote</code>	1068	<code>\text</code>	13, 14, 40
<code>\blx@info</code>	1172	<code>\textnormal</code>	40
<code>\blx@lasthash@foot</code>	1131	<code>\textsuperscript</code>	10
<code>\blx@lasthash@text</code>	1130	<code>\textup</code>	40
		<code>\thechapter</code>	28, 41, 61
		<code>\theHpostnote</code>	15
		<code>\thepage</code>	28, 105
		<code>\thepostnote</code>	17, 407, 471, 478
		<code>\thesection</code>	28, 44, 64

tl commands:	
\c_empty_tl	124
\tl_clear:N	71, 117, 888, 889, 890, 891, 892, 893, 1109, 1110, 1126, 1127, 1134, 1137, 1140, 1143
\tl_const:Nn	77, 291
\tl_gclear:N	577, 868, 869, 870, 871, 872, 873, 874, 875, 884, 885, 886, 887
\tl_gput_right:Nn	53
\tl_gset:Nn	981, 982, 986, 987, 990, 993, 994, 998, 999, 1002, 1005, 1006, 1010, 1011, 1014, 1017, 1018, 1022, 1023, 1026, 1155, 1760
\tl_gset_eq:NN	1168
\tl_if_empty:NTF	404, 466, 530, 532, 902, 918, 959, 1576, 1587, 1599, 1607
\tl_if_eq:NNTF	818, 904, 1050, 1162
\tl_if_eq:NnTF	645, 659, 686, 741
\tl_if_eq:nnTF	160, 896
\tl_new:N	16, 17, 147, 154, 221, 222, 223, 290, 294, 390, 427, 430, 509, 537, 584, 596, 597, 598, 599, 600, 601, 603, 604, 605, 606, 607, 608, 609, 610, 839, 840, 841, 842, 843, 844, 845, 846, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 1054, 1149, 1150, 1180, 1190, 1749, 1750, 1751, 1752, 1759
\tl_set:Nn	
 116, 163, 164, 168, 316, 391, 407, 471, 478, 515, 541, 626, 649, 650, 669, 681, 731, 732, 749, 953, 1111, 1112, 1113, 1114, 1153, 1194, 1647, 1666, 1700, 1718, 1753, 1754, 1755, 1756, 1778, 1779, 1780, 1781, 1794, 1795, 1796, 1797, 1806, 1807, 1808, 1809, 1816, 1817, 1818, 1819
	\togglefalse
	1108
	\toggletrue
	1063
	token commands:
	\token_to_str:N
	85, 93, 101
	\topsep
	186, 203, 1532, 1548
	U
	\undef
	1128, 1129, 1130, 1131
	\unkern
	11, 318, 319
	use commands:
	\use:N
	1101
	\use_none:n
	1171, 1172
	\UseHook
	52, 411, 463, 625, 695
	\UseInstance
	1523, 1539
	W
	\write
	1289, 1295
	Z
	\zcsetup
	1220, 1226
	\zlabel
	44, 48, 51, 533
	\zref
	1206, 1208

Local Variables: jinx-local-words: endnote endnotes End: