

# The zref-clever package implementation\*

Gustavo Barros<sup>†</sup>

2021-09-13

## Contents

|           |                                  |           |
|-----------|----------------------------------|-----------|
| <b>1</b>  | <b>Initial setup</b>             | <b>2</b>  |
| <b>2</b>  | <b>Dependencies</b>              | <b>2</b>  |
| <b>3</b>  | <b>zref setup</b>                | <b>2</b>  |
| <b>4</b>  | <b>Plumbing</b>                  | <b>7</b>  |
| 4.1       | Messages . . . . .               | 7         |
| 4.2       | Translations . . . . .           | 7         |
| 4.3       | Options . . . . .                | 9         |
| <b>5</b>  | <b>Type format</b>               | <b>20</b> |
| 5.1       | \zcRefTypeSetup . . . . .        | 20        |
| 5.2       | \zcDeclareTranslations . . . . . | 22        |
| <b>6</b>  | <b>\zceref</b>                   | <b>24</b> |
| <b>7</b>  | <b>\zcpageref</b>                | <b>25</b> |
| <b>8</b>  | <b>Sorting</b>                   | <b>25</b> |
| <b>9</b>  | <b>Typesetting</b>               | <b>33</b> |
| <b>10</b> | <b>Special handling</b>          | <b>54</b> |
| 10.1      | Appendix . . . . .               | 54        |
| 10.2      | \newtheorem . . . . .            | 55        |
| 10.3      | enumitem package . . . . .       | 55        |
| <b>11</b> | <b>Translations</b>              | <b>55</b> |
|           | <b>Index</b>                     | <b>70</b> |

---

\*This file describes v0.1.0-alpha, released 2021-09-13.

<sup>†</sup><https://github.com/gusbrs/zref-clever>

# 1 Initial setup

Start the DocStrip guards.

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

Taking a stance on backward compatibility of the package. During initial development, we have used freely recent features of the kernel (albeit refraining from `l3candidates`, even though I'd have loved to have used `\bool_case_true:...`). We presume `xparse` (which made to the kernel in the 2020-10-01 release), and `expl3` as well (which made to the kernel in the 2020-02-02 release). We also just use UTF-8 for the translations (which became the default input encoding in the 2018-04-01 release). Hence, since we would not be able to go much backwards without special handling anyway, we make the cut with the inclusion of the new hook management system (`ltxcmdhooks`), which is bound to be useful for our purposes, and was released with the 2021-06-01 kernel.

```
3 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
4 \IfFormatAtLeastTF{2021-06-01}
5 {}
6 {%
7   \PackageError{zref-clever}{LaTeX kernel too old}
8   {%
9     'zref-clever' requires a LaTeX kernel newer than 2021-06-01.%
10    \MessageBreak Loading will abort!%
11   }%
12   \endinput
13 }%
   Identify the package.
14 \ProvidesExplPackage {zref-clever} {2021-09-13} {0.1.0-alpha}
15 {Clever LaTeX cross-references based on zref}
```

# 2 Dependencies

Required packages. Besides these, `zref-hyperref` may also be required depending on the presence of `hyperref` itself and on the `hyperref` option.

```
16 \RequirePackage { zref-base }
17 \RequirePackage { zref-user }
18 \RequirePackage { zref-counter }
19 \RequirePackage { zref-abspage }
20 \RequirePackage { translations }
```

# 3 zref setup

For the purposes of the package, we need to store some information with the labels, some of it standard, some of it not so much. So, we have to setup `zref` to do so.

Some basic properties are handled by `zref` itself, or some of its modules. The `page` and `counter` properties are respectively provided by modules `zref-base` and `zref-counter`. The `zref-abspage` provides the `abspage` property which gives us a safe and easy way to sort labels for page references.

But the reference itself, stored by `zref-base` in the `default` property, is somewhat a disputed real estate. In particular, the use of `\labelformat` (previously from `varioref`, now in the kernel) will include there the reference “prefix” and complicate the job we are trying to do here. Hence, we isolate `\the\counter` and store it “clean” in `zc@thecnt` for reserved use. Based on the definition of `\@currentlabel` done inside `\refstepcounter` in ‘texdoc source2e’, section ‘ltxref.dtx’. We just drop the `\p@...` prefix.

```

21 \zref@newprop { zc@thecnt } { \use:c { the \@currentcounter } }
22 \zref@addprop \ZREF@mainlist { zc@thecnt }

```

Much of the work of `zref-clever` relies on the association between a label’s “counter” and its “type” (see the User manual section on “Reference types”). Superficially examined, one might think this relation could just be stored in a global property list, rather than in the label itself. However, there are cases in which we want to distinguish different types for the same counter, depending on the document context. Hence, we need to store the “type” of the “counter” for each “label”. In setting this, the presumption is that the label’s type has the same name as its counter, unless it is specified otherwise by the `countertype` option, as stored in `\l__zrefclever_counter_type_prop`.

```

23 \zref@newprop { zc@type }
24 {
25   \prop_if_in:NVTF \l__zrefclever_counter_type_prop \@currentcounter
26   {
27     \exp_args:NNe \prop_item:Nn
28     \l__zrefclever_counter_type_prop { \@currentcounter }
29   }
30   { \@currentcounter }
31 }
32 \zref@addprop \ZREF@mainlist { zc@type }

```

Since the `zc@thecnt` and `page` properties store the “*printed* representation” of their respective counters, for sorting and compressing purposes, we are also interested in their numeric values. So we store them in `zc@cntval` and `zc@pgval`. For this, we use `\c@<counter>`, which contains the counter’s numerical value (see ‘texdoc source2e’, section ‘ltxcounts.dtx’).

```

33 \zref@newprop { zc@cntval } [0] { \int_use:c { c@ \@currentcounter } }
34 \zref@addprop \ZREF@mainlist { zc@cntval }
35 \zref@newprop* { zc@pgval } [0] { \int_use:c { c@page } }
36 \zref@addprop \ZREF@mainlist { zc@pgval }

```

However, since many counters (may) get reset along the document, we require more than just their numeric values. We need to know the reset chain of a given counter, in order to sort and compress a group of references. Also here, the “printed representation” is not enough, not only because it is easier to work with the numeric values but, given we occasionally group multiple counters within a single type, sorting this group requires to know the actual counter reset chain (the counters’ names and values). Indeed, the set of counters grouped into a single type cannot be arbitrary: all of them must belong to the same reset chain, and must be nested within each other (they cannot even just share the same parent).

Furthermore, even if it is true that most of the definitions of counters, and hence of their reset behavior is likely to be defined in the preamble, this is not necessarily true. Users can create counters, newtheorems mid-document, and alter their reset behavior along the way. Was that not the case, we could just store the desired information at

`\begindocument` in a variable and retrieve it when needed. But since it is, we need to store the information with the label, with the values as current when the label is set.

Though counters can be reset at any time, and in different ways at that, the most important use case is the automatic resetting of counters when some other counter is stepped, as performed by the standard mechanisms of the kernel (optional argument of `\newcounter`, `\@addtoreset`, `\counterwithin` and related infrastructure). The canonical optional argument of `\newcounter` establishes that the counter being created (the mandatory argument) gets reset every time the “enclosing counter” gets stepped (this is called in the usual sources “within-counter”, “old counter”, “supercounter” etc.). This information is a little trickier to get. For starters, the counters which may reset the current counter are not retrievable from the counter itself, because this information is stored with the counter that does the resetting, not with the one that gets reset (the list is stored in `\cl@<counter>` with format `\@elt{countera}\@elt{counterb}\@elt{counterc}`, see section ‘ltcounts.dtx’ in ‘source2e’). Besides, there may be a chain of resetting counters, which must be taken into account: if ‘counterC’ gets reset by ‘counterB’, and ‘counterB’ gets reset by ‘counterA’, stepping the latter affects all three of them.

The procedure below examines a set of counters, those included in `\l__zrefclever_counter_resettters_seq`, and for each of them retrieves the set of counters it resets, as stored in `\cl@<counter>`, looking for the counter for which we are trying to set a label (`\@currentcounter`, passed as an argument to the functions). There is one relevant caveat to this procedure: `\l__zrefclever_counter_resettters_seq` is populated by hand with the “usual suspects”, there is no way (that I know of) to ensure it is exhaustive. However, it is not that difficult to create a reasonable “usual suspects” list which, of course, should include the counters for the sectioning commands to start with, and it is easy to add more counters to this list if needed, with the option `counterresetters`. Unfortunately, not all counters are created alike, or reset alike. Some counters, even some kernel ones, get reset by other mechanisms (notably, the `enumerate` environment counters do not use the regular counter machinery for resetting on each level, but are nested nevertheless by other means). Therefore, inspecting `\cl@<counter>` cannot possibly fully account for all of the automatic counter resetting which takes place in the document. And there’s also no other “general rule” we could grab on for this, as far as I know. So we provide a way to manually tell `zref-clever` of these cases, by means of the `counterresetby` option, whose information is stored in `\l__zrefclever_counter_resetby_prop`. This manual specification has precedence over the search through `\l__zrefclever_counter_resettters_seq`, and should be handled with care, since there is no possible verification mechanism for this.

```
\__zrefclever_get_enclosing_counters:n
__zrefclever_get_enclosing_counters_value:n
```

Recursively generate a *sequence* of “enclosing counters” and values, for a given `<counter>` and leave it in the input stream. These functions must be expandable, since they get called from `\zref@newprop` and are the ones responsible for generating the desired information when the label is being set. Note that the order in which we are getting this information is reversed, since we are navigating the counter reset chain bottom-up. But it is very hard to do otherwise here where we need expandable functions, and easy to handle at the reading side.

```
\__zrefclever_get_enclosing_counters:n {<counter>}
\__zrefclever_get_enclosing_counters_value:n {<counter>}

37 \cs_new:Npn \__zrefclever_get_enclosing_counters:n #1
38 {
39   \cs_if_exist:cT { c@ \__zrefclever_counter_reset_by:n {#1} }
```

```

40     {
41       { \_zrefclever_counter_reset_by:n {#1} }
42       \_zrefclever_get_enclosing_counters:e
43       { \_zrefclever_counter_reset_by:n {#1} }
44     }
45   }
46   \cs_new:Npn \_zrefclever_get_enclosing_counters_value:n #1
47   {
48     \cs_if_exist:cT { c@ \_zrefclever_counter_reset_by:n {#1} }
49     {
50       { \int_use:c { c@ \_zrefclever_counter_reset_by:n {#1} } }
51       \_zrefclever_get_enclosing_counters_value:e
52       { \_zrefclever_counter_reset_by:n {#1} }
53     }
54   }

```

Both `e` and `f` expansions work for this particular recursive call. For the time being, I'll stay with the `e` variant, since conceptually it is what I want (`x` itself is not expandable), and this package is unlikely to be used within the context of older kernels for which the performance penalty of the `e` expansion would ensue (see also [https://tex.stackexchange.com/q/611370/#comment1529282\\_611385](https://tex.stackexchange.com/q/611370/#comment1529282_611385), thanks Enrico Gregorio, aka ‘egreg’).

```

55 \cs_generate_variant:Nn \_zrefclever_get_enclosing_counters:n { V , e }
56 \cs_generate_variant:Nn \_zrefclever_get_enclosing_counters_value:n { V , e }

```

(End definition for `\_zrefclever_get_enclosing_counters:n` and `\_zrefclever_get_enclosing_counters_value:n`.)

`\_zrefclever_counter_reset_by:n` Auxiliary function for `\_zrefclever_get_enclosing_counters:n` and `\_zrefclever_get_enclosing_counters_value:n`. They are broken in parts to be able to use the expandable mapping functions. `\_zrefclever_counter_reset_by:n` leaves in the stream the “enclosing counter” which resets `\counter`.

```

    \_zrefclever_counter_reset_by:n {<counter>}

57 \cs_new:Npn \_zrefclever_counter_reset_by:n #1
58   {
59     \bool_if:nTF
60       { \prop_if_in_p:Nn \l__zrefclever_counter_resetby_prop {#1} }
61       { \prop_item:Nn \l__zrefclever_counter_resetby_prop {#1} }
62       {
63         \seq_map_tokens:Nn \l__zrefclever_counter_resettters_seq
64         { \_zrefclever_counter_reset_by_aux:nn {#1} }
65       }
66   }
67 \cs_new:Npn \_zrefclever_counter_reset_by_aux:nn #1#2
68   {
69     \cs_if_exist:cT { c@ #2 }
70     {
71       \tl_if_empty:cF { c1@ #2 }
72       {
73         \tl_map_tokens:cn { c1@ #2 }
74         { \_zrefclever_counter_reset_by_auxi:nnn {#2} {#1} }
75       }
76   }

```

```

77 }
78 \cs_new:Npn \__zrefclever_counter_reset_by_auxi:nnn #1#2#3
79 {
80   \str_if_eq:nnT {#2} {#3}
81   { \tl_map_break:n { \seq_map_break:n {#1} } }
82 }

```

(End definition for `\__zrefclever_counter_reset_by:n`.)

Finally, we create the `zc@enclcnt` and `zc@enclval` properties, and add them to the main property list.

```

83 \zref@newprop { zc@enclcnt }
84 { \__zrefclever_get_enclosing_counters:V \@currentcounter }
85 \zref@newprop { zc@enclval }
86 { \__zrefclever_get_enclosing_counters_value:V \@currentcounter }
87 \zref@addprop \ZREF@mainlist { zc@enclcnt }
88 \zref@addprop \ZREF@mainlist { zc@enclval }

```

Another piece of information we need is the page numbering format being used by `\thepage`, so that we know when we can (or not) group a set of page references in a range. Unfortunately, `page` is not a typical counter in ways which complicates things. First, it does commonly get reset along the document, not necessarily by the usual counter reset chains, but rather with `\pagenumbering` or variations thereof. Second, the format of the page number commonly changes in the document (roman, arabic, etc.), not necessarily, though usually, together with a reset. Trying to “parse” `\thepage` to retrieve such information is bound to go wrong: we don’t know, and can’t know, what is within that macro, and that’s the business of the user, or of the documentclass, or of the loaded packages. The technique used by `cleveref`, which we borrow here, is simple and smart: store with the label what `\thepage` would return, if the counter `\c@page` was “1”. That does not allow us to *sort* the references, luckily however, we have `abspage` which solves this problem. But we can decide whether two labels can be compressed into a range or not based on this format: if they are identical, we can compress them, otherwise, we can’t. To do so, we locally redefine `\c@page` to return “1”, thus avoiding any global spillovers of this trick. Since this operation is not expandable we cannot run it directly from the property definition. Hence, we use a shipout hook, and set `\g__zrefclever_page_format_tl`, which can then be retrieved by the starred definition of `\zref@newprop*{zc@pgfmt}`.

```

89 \tl_new:N \g__zrefclever_page_format_tl
90 \cs_new_protected:Npx \__zrefclever_page_format_aux: { \int_eval:n { 1 } }
91 \AddToHook { shipout / before }
92 {
93   \group_begin:
94   \cs_set_eq:NN \c@page \__zrefclever_page_format_aux:
95   \exp_args:NNx \tl_gset:Nn \g__zrefclever_page_format_tl { \thepage }
96   \group_end:
97 }
98 \zref@newprop* { zc@pgfmt } { \g__zrefclever_page_format_tl }
99 \zref@addprop \ZREF@mainlist { zc@pgfmt }

```

Still another property which we don’t need to handle at the data provision side, but need to cater for at the retrieval side, is the `url` property (or the equivalent `urluse`) from the `zref-xr` module, which is added to the labels imported from external documents, and needed to construct hyperlinks to them.

## 4 Plumbing

### 4.1 Messages

```

100 \msg_new:nnn { zref-clever } { option-not-type-specific }
101 {
102   Option~'#1'~is-not-type-specific~\msg_line_context:..~
103   Set~it~in~'\exp_not:N \zcDeclareTranslations'~before-first~'type'~switch~
104   or~as~package~option.
105 }
106 \msg_new:nnn { zref-clever } { option-only-type-specific }
107 {
108   No~type~specified~for~option~'#1'~\msg_line_context:..~
109   Set~it~after~'type'~switch~or~in~'\exp_not:N \zcRefTypeSetup'.
110 }
111 \msg_new:nnn { zref-clever } { key-requires-value }
112 { The~'#1'~key~'#2'~requires~a~value. }
113 \msg_new:nnn { zref-clever } { missing-zref-titleref }
114 {
115   Option~'ref=title'~requested~\msg_line_context:..~
116   But~package~'zref-titleref'~is~not~loaded,~falling-back~to~default~'ref'.
117 }
118 \msg_new:nnn { zref-clever } { hyperref-preamble-only }
119 {
120   Option~'hyperref'~only~available~in~the~preamble. \iow_newline:
121   Use~the~starred~version~of~'\noexpand\zcheck'~instead.
122 }
123 \msg_new:nnn { zref-clever } { missing-hyperref }
124 { Missing~'hyperref'~package. \iow_newline: Setting~'hyperref=false'. }
125 \msg_new:nnn { zref-check } { check-document-only }
126 { Option~'check'~only~available~in~the~document. }
127 \msg_new:nnn { zref-clever } { missing-zref-check }
128 {
129   Option~'check'~requested~\msg_line_context:..~
130   But~package~'zref-check'~is~not~loaded,~can't~run~the~checks.
131 }
132 \msg_new:nnn { zref-clever } { counters-not-nested }
133 { Counters~not~nested~for~labels~'#1'~and~'#2'~\msg_line_context:.. }
134 \msg_new:nnn { zref-clever } { missing-type }
135 { Reference~type~undefined~for~label~'#1'~\msg_line_context:.. }
136 \msg_new:nnn { zref-clever } { missing-name }
137 { Name~undefined~for~type~'#1'~\msg_line_context:.. }
138 \msg_new:nnn { zref-clever } { single-element-range }
139 { Range~for~type~'#1'~resulted~in~single~element~\msg_line_context:.. }

```

### 4.2 Translations

Some wrappers around translations functions, so that we can generate variants with expansion control for arguments, or for convenience.

`\_zrefclever_if_transl:nnTF` Conditional to check if a translation of  $\langle key \rangle$  exists for language  $\langle lang \rangle$ .

`\_zrefclever_if_transl:nnTF { $\langle lang \rangle$ } { $\langle key \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }`

140 `\prg_new_conditional:Npnn \_zrefclever_if_transl:nn #1#2 { TF }`

```

141 {
142   \IfTranslation {#1} {#2}
143   { \prg_return_true: }
144   { \prg_return_false: }
145 }
146 \prg_generate_conditional_variant:Nnn \__zrefclever_if_transl:nn { xx } { TF }
(End definition for \__zrefclever_if_transl:nnTF.)

```

`\__zrefclever_get_transl:nnn` Retrieves the translation of  $\langle key \rangle$  for the language  $\langle lang \rangle$  and saves it in  $\langle macro \rangle$ .

```

\__zrefclever_get_transl:nnn {<macro>} {<lang>} {<key>}
147 \cs_new_protected:Npn \__zrefclever_get_transl:nnn #1#2#3
148 { \SaveTranslationFor{#1}{#2}{#3} }
149 \cs_generate_variant:Nn \__zrefclever_get_transl:nnn { nxx }
(End definition for \__zrefclever_get_transl:nnn.)

```

`\__zrefclever_declare_transl:nnn` Defines the translation of  $\langle key \rangle$  for the language  $\langle lang \rangle$ . The  $\langle key \rangle$  here is the full key, including package prefix, type, and internal key name (i.e. the “key” from the perspective of translations).

```

\__zrefclever_declare_transl:nnn {<lang>} {<key>} {<translation>}
150 \cs_new_protected:Npn \__zrefclever_declare_transl:nnn #1#2#3
151 { \declaretranslation {#1} {#2} {#3} }
152 \cs_generate_variant:Nn \__zrefclever_declare_transl:nnn { xxn }
(End definition for \__zrefclever_declare_transl:nnn.)

```

`\__zrefclever_declare_default_transl:nnn` Defines the translation of  $\langle key \rangle$  for the language  $\langle lang \rangle$ . The  $\langle key \rangle$  here is the internal key name (i.e. the name of the option).

```

\__zrefclever_declare_default_transl:nnn {<lang>} {<key>} {<translation>}
153 \cs_new_protected:Npn \__zrefclever_declare_default_transl:nnn #1#2#3
154 { \declaretranslation {#1} { zrefclever-default- #2 } {#3} }
(End definition for \__zrefclever_declare_default_transl:nnn.)

```

`\zcDicDefaultTransl` Functions for providing translations in dictionary files. We refrain from using `expl3` names and “atletter”, so that we don’t have to control catcodes in those files (as far as I can tell, translations itself doesn’t cater for this), even if these commands are only really meant for internal use. The  $\langle key \rangle$  here is always the internal key name (i.e. the name of the option). The language does not need to be specified, it is automatically retrieved from the dictionary’s declaration done by `\ProvideDictionaryFor`. Since `\ProvideDictTranslation` is restricted by translations to the preamble, we inherit this restriction here.

```

\zcDicDefaultTransl {<key>} {<translation>}
\zcDicTypeTransl {<type>} {<key>} {<translation>}
155 \NewDocumentCommand \zcDicDefaultTransl { m m }
156 { \ProvideDictTranslation { zrefclever-default- #1 } {#2} }
157 \NewDocumentCommand \zcDicTypeTransl { m m m }
158 { \ProvideDictTranslation { zrefclever-type- #1 - #2 } {#3} }
159 \@onlypreamble \zcDicDefaultTransl
160 \@onlypreamble \zcDicTypeTransl
(End definition for \zcDicDefaultTransl and \zcDicTypeTransl.)

```



## 4.3 Options

### Auxiliary functions

`\_zrefclever_prop_put_non_empty:Nnn` If  $\langle value \rangle$  is empty, remove  $\langle key \rangle$  from  $\langle property list \rangle$ . Otherwise, add  $\langle key \rangle = \langle value \rangle$  to  $\langle property list \rangle$ .

```

\__zrefclever_prop_put_non_empty:Nnn <property list> {<key>} {<value>}

161 \cs_new_protected:Npn \__zrefclever_prop_put_non_empty:Nnn #1#2#3
162 {
163   \tl_if_empty:nTF {#3}
164     { \prop_remove:Nn #1 {#2} }
165     { \prop_put:Nnn #1 {#2} {#3} }
166 }
```

(End definition for `\_zrefclever_prop_put_non_empty:Nnn`.)

### countertype option

`\l__zrefclever_counter_type_prop` is used by `zc@type` property, and stores a mapping from “counter” to “reference type”. Only those counters whose type name is different from that of the counter need to be specified, since `zc@type` presumes the counter as the type if the counter is not found in `\l__zrefclever_counter_type_prop`.

```

167 \prop_new:N \l__zrefclever_counter_type_prop
168 \keys_define:nn { zref-clever }
169 {
170   countertype .code:n =
171   {
172     \keyval_parse:nnn
173     {
174       \msg_warning:nnnn { zref-clever }
175       { key-requires-value } { countertype }
176     }
177     {
178       \__zrefclever_prop_put_non_empty:Nnn
179       \l__zrefclever_counter_type_prop
180     }
181     {#1}
182   } ,
183   countertype .value_required:n = true ,
184   countertype .initial:n =
185   {
186     subsection    = section ,
187     subsubsection = section ,
188     subparagraph  = paragraph ,
189     enumi         = item ,
190     enumii        = item ,
191     enumiii       = item ,
192     enumiv        = item ,
193   } ,
194 }
```

### counterresetters option

`\l__zrefclever_counter_resetters_seq` is used by `\__zrefclever_counter_reset_by:n` to populate the `zc@enclcnt` and `zc@enclval` properties, and stores the list of counters which are potential “enclosing counters” for other counters. This option is constructed such that users can only *add* items to the variable. There would be little gain and some risk in allowing removal, and the syntax of the option would become unnecessarily more complicated. Besides, users can already override, for any particular counter, the search done from the set in `\l__zrefclever_counter_resetters_seq` with the `counterresetby` option.

```
195 \seq_new:N \l__zrefclever_counter_resetters_seq
196 \keys_define:nn { zref-clever }
197 {
198   counterresetters .code:n =
199   {
200     \clist_map_inline:nn {#1}
201     {
202       \seq_if_in:NnF \l__zrefclever_counter_resetters_seq {##1}
203       {
204         \seq_put_right:Nn
205         \l__zrefclever_counter_resetters_seq {##1}
206       }
207     }
208   } ,
209   counterresetters .initial:n =
210   {
211     part ,
212     chapter ,
213     section ,
214     subsection ,
215     subsubsection ,
216     paragraph ,
217     subparagraph ,
218   },
219   typesort .value_required:n = true ,
220 }
```

### counterresetby option

`\l__zrefclever_counter_resetby_prop` is used by `\__zrefclever_counter_reset_by:n` to populate the `zc@enclcnt` and `zc@enclval` properties, and stores a mapping from counters to the counter which resets each of them. This mapping has precedence in `\__zrefclever_counter_reset_by:n` over the search through `\l__zrefclever_counter_resetters_seq`.

```
221 \prop_new:N \l__zrefclever_counter_resetby_prop
222 \keys_define:nn { zref-clever }
223 {
224   counterresetby .code:n =
225   {
226     \keyval_parse:nnn
227     {
228       \msg_warning:nnn { zref-clever }
```

```

229         { key-requires-value } { counterresetby }
230     }
231     {
232         \__zrefclever_prop_put_non_empty:Nnn
233         \l__zrefclever_counter_resetby_prop
234     }
235     {#1}
236 } ,
237 counterresetby .value_required:n = true ,
238 counterresetby .initial:n =
239 {

```

The counters for the `enumerate` environment do not use the regular counter machinery for resetting on each level, but are nested nevertheless by other means, treat them as exception.

```

240     enumii = enumi ,
241     enumiii = enumii ,
242     enumiv = enumiii ,
243 } ,
244 }

```

## ref option

Stores whether this reference is to the page, or to the default counter.

```

245 \tl_new:N \l__zrefclever_ref_property_tl
246 \bool_new:N \l__zrefclever_page_ref_bool
247 \keys_define:nn { zref-clever }
248 {
249     ref .choice: ,
250     ref / zc@thecnt .code:n =
251     {
252         \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
253         \bool_set_false:N \l__zrefclever_page_ref_bool
254     } ,
255     ref / page .code:n =
256     {
257         \tl_set:Nn \l__zrefclever_ref_property_tl { page }
258         \bool_set_true:N \l__zrefclever_page_ref_bool
259     } ,
260     ref / title .code:n =
261     {
262         \AddToHook { begindocument }
263         {
264             \@ifpackageloaded { zref-titleref }
265             {
266                 \tl_set:Nn \l__zrefclever_ref_property_tl { title }
267                 \bool_set_false:N \l__zrefclever_page_ref_bool
268             }
269             {
270                 \msg_warning:nn { zref-clever } { missing-zref-titleref }
271                 \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
272                 \bool_set_false:N \l__zrefclever_page_ref_bool
273             }
274         }

```

```

275     } ,
276     ref .initial:n = zc@thecnt ,
277     ref .value_required:n = true ,
278     page .meta:n = { ref = page },
279     page .value_forbidden:n = true ,
280 }
281
282 \AddToHook { begindocument }
283 {
284   \@ifpackageloaded { zref-titleref }
285   {
286     \keys_define:nn { zref-clever }
287     {
288       ref / title .code:n =
289       {
290         \tl_set:Nn \l__zrefclever_ref_property_tl { title }
291         \bool_set_false:N \l__zrefclever_page_ref_bool
292       }
293     }
294   }
295   {
296     \keys_define:nn { zref-clever }
297     {
298       ref / title .code:n =
299       {
300         \msg_warning:nn { zref-clever } { missing-zref-titleref }
301         \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
302         \bool_set_false:N \l__zrefclever_page_ref_bool
303       }
304     }
305   }
306 }

```

Currently, we restrict ‘ref=’ to these two (or three) alternatives, but there might be a case for making this more flexible. The infrastructure can already handle receiving an arbitrary property, as long as one is satisfied with sorting and compressing from the default counter. If more flexibility is granted, one thing *must* be handled at this point: the existence of the property itself, as far as `zref` is concerned. This because typesetting relies on the check `\zref@ifrefcontainsprop`, which *presumes* the property is defined and silently expands the *true* branch if it is not (see <https://github.com/ho-tex/zref/issues/13>, thanks Ulrike Fischer). Therefore, before adding anything to `\l__zrefclever_ref_property_tl`, check if first here with `\zref@ifpropundefined`: close it at the door.

### typeset option

```

307 \bool_new:N \l__zrefclever_typeset_ref_bool
308 \bool_new:N \l__zrefclever_typeset_name_bool
309 \keys_define:nn { zref-clever }
310 {
311   typeset .choice: ,
312   typeset / both .code:n =
313   {
314     \bool_set_true:N \l__zrefclever_typeset_ref_bool

```

```

315         \bool_set_true:N \l__zrefclever_typeset_name_bool
316     } ,
317     typeset / ref .code:n =
318     {
319         \bool_set_true:N \l__zrefclever_typeset_ref_bool
320         \bool_set_false:N \l__zrefclever_typeset_name_bool
321     } ,
322     typeset / name .code:n =
323     {
324         \bool_set_false:N \l__zrefclever_typeset_ref_bool
325         \bool_set_true:N \l__zrefclever_typeset_name_bool
326     } ,
327     typeset .initial:n = both ,
328     typeset .value_required:n = true ,
329
330     noname .meta:n = { typeset = ref } ,
331     noname .value_forbidden:n = true ,
332 }

```

### sort option

User option, sort labels ranges or not

```

333 \bool_new:N \l__zrefclever_typeset_sort_bool
334 \keys_define:nn { zref-clever }
335 {
336     sort .bool_set:N = \l__zrefclever_typeset_sort_bool ,
337     sort .initial:n = true ,
338     sort .default:n = true ,
339     nosort .meta:n = { sort = false } ,
340     nosort .value_forbidden:n = true ,
341 }

```

### typesort option

```

342 \seq_new:N \l__zrefclever_typesort_seq
343 \keys_define:nn { zref-clever }
344 {
345     typesort .code:n =
346     {
347         \seq_set_from_clist:Nn \l__zrefclever_typesort_seq {#1}
348         % Reverse the sequence, since the sort priorities are computed in the
349         % negative range, so that we can implicitly rely on '0' being the
350         % 'last value'.
351         \seq_reverse:N \l__zrefclever_typesort_seq
352     } ,
353     typesort .initial:n =
354     { part , chapter , section , paragraph } ,
355     typesort .value_required:n = true ,
356     notypesort .code:n =
357     { \seq_clear:N \l__zrefclever_typesort_seq } ,
358     notypesort .value_forbidden:n = true ,
359 }

```

### comp option

User option, compress ranges or not

```
360 \bool_new:N \l__zrefclever_typeset_compress_bool
361 \keys_define:nn { zref-clever }
362 {
363   comp .bool_set:N = \l__zrefclever_typeset_compress_bool ,
364   comp .initial:n = true ,
365   comp .default:n = true ,
366   nocomp .meta:n = { comp = false },
367   nocomp .value_forbidden:n = true ,
368 }
```

### range option

```
369 \bool_new:N \l__zrefclever_typeset_range_bool
370 \keys_define:nn { zref-clever }
371 {
372   range .bool_set:N = \l__zrefclever_typeset_range_bool ,
373   range .initial:n = false ,
374   range .default:n = true ,
375 }
```

### hyperref option

`\l__zrefclever_use_hyperref_bool`  
`\l__zrefclever_warn_hyperref_bool`

```
376 \bool_new:N \l__zrefclever_use_hyperref_bool
377 \bool_new:N \l__zrefclever_warn_hyperref_bool
378 \keys_define:nn { zref-clever }
379 {
380   hyperref .choice: ,
381   hyperref / auto .code:n =
382   {
383     \bool_set_true:N \l__zrefclever_use_hyperref_bool
384     \bool_set_false:N \l__zrefclever_warn_hyperref_bool
385   } ,
386   hyperref / true .code:n =
387   {
388     \bool_set_true:N \l__zrefclever_use_hyperref_bool
389     \bool_set_true:N \l__zrefclever_warn_hyperref_bool
390   } ,
391   hyperref / false .code:n =
392   {
393     \bool_set_false:N \l__zrefclever_use_hyperref_bool
394     \bool_set_false:N \l__zrefclever_warn_hyperref_bool
395   } ,
396   hyperref .initial:n = auto ,
397   hyperref .default:n = auto
398 }
```

*(End definition for \l\_\_zrefclever\_use\_hyperref\_bool and \l\_\_zrefclever\_warn\_hyperref\_bool.)*

```
399 \AddToHook { begindocument }
400 {
401   \@ifpackageloaded { hyperref }
402   {
```

```

403     \bool_if:NT \l__zrefclever_use_hyperref_bool
404     { \RequirePackage { zref-hyperref } }
405   }
406   {
407     \bool_if:NT \l__zrefclever_warn_hyperref_bool
408     { \msg_warning:nn { zref-clever } { missing-hyperref } }
409     \bool_set_false:N \l__zrefclever_use_hyperref_bool
410   }
411   \keys_define:nn { zref-clever }
412   {
413     hyperref .code:n =
414     { \msg_warning:nn { zref-clever } { hyperref-preamble-only } }
415   }
416 }

```

### nameinlink option

\l\_\_zrefclever\_nameinlink\_tl

```

417 \str_new:N \l__zrefclever_nameinlink_str
418 \keys_define:nn { zref-clever }
419 {
420   nameinlink .choice: ,
421   nameinlink / true .code:n =
422   { \str_set:Nn \l__zrefclever_nameinlink_str { true } } ,
423   nameinlink / false .code:n =
424   { \str_set:Nn \l__zrefclever_nameinlink_str { false } } ,
425   nameinlink / single .code:n =
426   { \str_set:Nn \l__zrefclever_nameinlink_str { single } } ,
427   nameinlink / tsingle .code:n =
428   { \str_set:Nn \l__zrefclever_nameinlink_str { tsingle } } ,
429   nameinlink .initial:n = tsingle ,
430   nameinlink .default:n = true ,
431 }

```

(End definition for \l\_\_zrefclever\_nameinlink\_tl.)

### cap capfirst options

```

432 \bool_new:N \l__zrefclever_capitalize_bool
433 \bool_new:N \l__zrefclever_capitalize_first_bool
434 \keys_define:nn { zref-clever }
435 {
436   cap .bool_set:N = \l__zrefclever_capitalize_bool ,
437   cap .initial:n = false ,
438   cap .default:n = true ,
439   nocap .meta:n = { cap = false } ,
440   nocap .value_forbidden:n = true ,
441
442   capfirst .bool_set:N = \l__zrefclever_capitalize_first_bool ,
443   capfirst .initial:n = false ,
444   capfirst .default:n = true ,
445
446   C .meta:n =
447   { capfirst = true , noabbrevfirst = true } ,

```

```

448     C .value_forbidden:n = true ,
449 }

```

#### abbrev noabbrevfirst option

```

450 \bool_new:N \l__zrefclever_abbrev_bool
451 \bool_new:N \l__zrefclever_noabbrev_first_bool
452 \keys_define:nn { zref-clever }
453 {
454     abbrev .bool_set:N = \l__zrefclever_abbrev_bool ,
455     abbrev .initial:n = false ,
456     abbrev .default:n = true ,
457     noabbrev .meta:n = { abbrev = false },
458     noabbrev .value_forbidden:n = true ,
459
460     noabbrevfirst .bool_set:N = \l__zrefclever_noabbrev_first_bool ,
461     noabbrevfirst .initial:n = false ,
462     noabbrevfirst .default:n = true ,
463 }

```

#### lang option

```

464 \tl_new:N \l__zrefclever_ref_language_tl
465 \tl_new:N \l__zrefclever_main_language_tl
466 \tl_new:N \l__zrefclever_current_language_tl
467 \NewHook { zref-clever / reflanguage }
468 \keys_define:nn { zref-clever }
469 {
470     lang .code:n =
471     {
472         \AddToHook { zref-clever / reflanguage }
473         {
474             \str_case:nnF {#1}
475             {
476                 { main }
477                 {
478                     \tl_set_eq:NN
479                     \l__zrefclever_ref_language_tl \l__zrefclever_main_language_tl
480                 }
481
482                 { current }
483                 {
484                     \tl_set_eq:NN
485                     \l__zrefclever_ref_language_tl \l__zrefclever_current_language_tl
486                 }
487             }
488         {
489             \tl_set:Nn \l__zrefclever_ref_language_tl {#1}
490             % If user specified a language at the preamble, make sure it
491             % is loaded.
492             \exp_args:Nx \file_if_exist:nTF
493             { zref-clever- \@trnslt@language {#1} .trsl }
494             { \LoadDictionaryFor {#1} { zref-clever } }
495             {
496                 \exp_args:Nx \file_if_exist:nT
497                 { zref-clever- \baselanguage {#1} .trsl }

```



```

498             { \LoadDictionaryFor {#1} { zref-clever } }
499         }
500     }
501 }
502 },
503 lang .initial:n = main ,
504 lang .value_required:n = true ,
505 }

\AtEndOfPackage so that it comes after \ProcessKeysOptions.
506 \AtEndOfPackage
507 {
508     \AddToHook { zref-clever / reflanguage }
509     {
510         \keys_define:nn { zref-clever }
511         {
512             lang .code:n =
513             {
514                 \str_case:nnF {#1}
515                 {
516                     { main }
517                     {
518                         \tl_set_eq:NN
519                         \l__zrefclever_ref_language_tl \l__zrefclever_main_language_tl
520                     }
521                     { current }
522                     {
523                         \tl_set_eq:NN
524                         \l__zrefclever_ref_language_tl \l__zrefclever_current_language_tl
525                     }
526                 }
527             }
528             { \tl_set:Nn \l__zrefclever_ref_language_tl {#1} }
529         } ,
530         lang .value_required:n = true ,
531     }
532 }
533 }

```

See <https://tex.stackexchange.com/a/233178> (including Javier Bezos' comment). Also <https://tex.stackexchange.com/a/281220> (including PLK's comments).

```

534 \AddToHook { begindocument / before }
535 {
536     % An internal alias for \pkg{translations}'s internal macro
537     % \cs{@trnslt@current@language}.
538     \tl_set_eq:NN \l__zrefclever_current_language_tl \@trnslt@current@language
539     % Getting main languages and, for each babel/polyglossia loaded language,
540     % load corresponding zref-clever dictionary.
541     \@ifpackageloaded{babel}
542     {
543         \tl_set_eq:NN \l__zrefclever_main_language_tl \bbl@main@language
544         \clist_map_inline:Nn \bbl@loaded
545         {
546             % Funny enough, \pkg{translations} also loads its basic
547             % dictionaries for all languages loaded by babel or polyglossia.

```

```

548 % First, there is no way to disable this, even if we don't need
549 % them at all here. Second, \pkg{translations} sends messages of
550 % its own missing dictionaries to 'info' and everyone else's to
551 % 'warning'... So we have to control ourselves for missing
552 % dictionaries and load them only if available.
553 \exp_args:Nx \file_if_exist:nTF
554 { zref-clever- \@trnslt@language {#1} .trsl }
555 { \LoadDictionaryFor {#1} { zref-clever } }
556 {
557   \exp_args:Nx \file_if_exist:nT
558   { zref-clever- \baselanguage {#1} .trsl }
559   { \LoadDictionaryFor {#1} { zref-clever } }
560 }
561 }
562 }
563 {
564   \@ifpackageloaded{polyglossia}
565   {
566     \tl_set_eq:NN \l__zrefclever_main_language_tl \xpg@main@language
567     \clist_map_inline:Nn \xpg@loaded
568     {
569       \exp_args:Nx \file_if_exist:nTF
570       { zref-clever- \@trnslt@language {#1} .trsl }
571       { \LoadDictionaryFor {#1} { zref-clever } }
572       {
573         \exp_args:Nx \file_if_exist:nT
574         { zref-clever- \baselanguage {#1} .trsl }
575         { \LoadDictionaryFor {#1} { zref-clever } }
576       }
577     }
578   }
579   {
580     \tl_set:Nn \l__zrefclever_main_language_tl { english }
581     \LoadDictionaryFor { english } { zref-clever }
582   }
583 }
584 % *Then* we execute the package options stored in the 'reflanguage' hook.
585 \UseHook { zref-clever / reflanguage }
586 }

```

#### note option

```

587 \tl_new:N \l__zrefclever_zceref_note_tl
588 \keys_define:nn { zref-clever }
589 {
590   note .tl_set:N = \l__zrefclever_zceref_note_tl ,
591   note .value_required:n = true ,
592 }

```

#### check option

Integration with zref-check.

```

593 \bool_new:N \l__zrefclever_zrefcheck_available_bool
594 \bool_new:N \l__zrefclever_zceref_with_check_bool
595 \keys_define:nn { zref-clever }
596 {

```

```

597     check .code:n =
598       { \msg_warning:nn { zref-clever } { check-document-only } } ,
599   }
600 \AddToHook { begindocument }
601 {
602   \@ifpackageloaded { zref-check }
603   {
604     \bool_set_true:N \l__zrefclever_zrefcheck_available_bool
605     \keys_define:nn { zref-clever }
606     {
607       check .code:n =
608       {
609         \bool_set_true:N \l__zrefclever_zcref_with_check_bool
610         \keys_set:nn { zref-check / zcheck } {#1}
611       }
612     }
613   }
614   {
615     \bool_set_false:N \l__zrefclever_zrefcheck_available_bool
616     \keys_define:nn { zref-clever }
617     {
618       check .code:n =
619       { \msg_warning:nn { zref-clever } { missing-zref-check } }
620     }
621   }
622 }

```

## Reference options

```

623 \tl_new:N \l__zrefclever_ref_typeset_font_tl
624 \keys_define:nn { zref-clever }
625 { font .tl_set:N = \l__zrefclever_ref_typeset_font_tl }

```

Only not necessarily type-specific options are pertinent here.

```

626 \prop_new:N \l__zrefclever_ref_options_prop
627 \clist_map_inline:nn
628 {
629   % Not type-specific options.
630   tpairsep ,
631   tlistsep ,
632   tlastsep ,
633   notesep ,
634   % Possibly type-specific options.
635   namefont ,
636   namesep ,
637   pairsep ,
638   listsep ,
639   lastsep ,
640   rangesep ,
641   reffont ,
642   refpre ,
643   refpos ,
644   reffont-in ,
645   refpre-in ,
646   refpos-in ,

```

```

647 }
648 {
649   \keys_define:nn { zref-clever }
650   {
651     #1 .default:V = \c_novalue_tl ,
652     #1 .code:n =
653     {
654       \tl_if_novalue:nTF {##1}
655       { \prop_remove:Nn \l__zrefclever_ref_options_prop {#1} }
656       { \prop_put:Nnn \l__zrefclever_ref_options_prop {#1} {##1} }
657     } ,
658   }
659 }

```

### Package options

Process load-time package options (<https://tex.stackexchange.com/a/15840>).

```

660 \RequirePackage { l3keys2e }
661 \ProcessKeysOptions { zref-clever }

```

`\zcsetup` Provide `\zcsetup`.

```

662 \NewDocumentCommand \zcsetup { m }
663 { \keys_set:nn { zref-clever } {#1} }

```

(End definition for `\zcsetup`.)

## 5 Type format

### 5.1 `\zcRefTypeSetup`

`\l__zrefclever_setup_type_tl` Variables storing the language and type to be used in `\zcRefTypeSetup` and `\zcDeclareTranslations`.

```

\l__zrefclever_setup_language_tl
664 \tl_new:N \l__zrefclever_setup_type_tl
665 \tl_new:N \l__zrefclever_setup_language_tl

```

(End definition for `\l__zrefclever_setup_type_tl` and `\l__zrefclever_setup_language_tl`.)

`\zcRefTypeSetup` Provide `\zcRefTypeSetup`.

```

666 \NewDocumentCommand \zcRefTypeSetup { m m }
667 {
668   \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
669   { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
670   \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
671   \keys_set:nn { zref-clever / typesetup } {#2}
672 }

```

(End definition for `\zcRefTypeSetup`.)

Inside `\zcRefTypeSetup` any of the options *can* receive empty values, and those values, if they exist in the property list, will override translations, regardless of their emptiness. In principle, we could live with the situation of, once a setting has made `\l__zrefclever_type_<type>_options_prop` or `\l__zrefclever_ref_options_prop` it stays there forever, and can only be overridden by a new value at the same precedence level or a higher one. But it would be nice if an user can “unset” an option at either of those to go back to the lower precedence level of the translations at any given point. So both in `\zcRefTypeSetup` and in setting reference options, we leverage the distinction

of an “empty valued key” (`key=` or `key=`) from a “key with no value” (`key`). This distinction is captured internally by the lower-level key parsing, but must be made explicit at `\keys_set:nn` by means of the `.default:` property of the key in `\keys_define:nn`. For the technique, see <https://tex.stackexchange.com/q/614690> (thanks Jonathan P. Spratte, aka ‘Skillmon’, and Phelype Oleinik).

Not type-specific options.

```

673 \clist_map_inline:nn
674 {
675     tpairsep ,
676     tlistsep ,
677     tlastsep ,
678     notesep ,
679 }
680 {
681     \keys_define:nn { zref-clever / typesetup }
682     {
683         #1 .code:n =
684         {
685             \msg_warning:nnn { zref-clever } { option-not-type-specific } {#1}
686         } ,
687     }
688 }

```

Possibly or necessarily type-specific options.

```

689 \clist_map_inline:nn
690 {
691     % Possibly type-specific options.
692     namefont ,
693     namesep ,
694     pairsep ,
695     listsep ,
696     lastsep ,
697     rangesep ,
698     reffont ,
699     refpre ,
700     refpos ,
701     reffont-in ,
702     refpre-in ,
703     refpos-in ,
704     % Necessarily type-specific options.
705     Name-sg ,
706     name-sg ,
707     Name-pl ,
708     name-pl ,
709     Name-sg-ab ,
710     name-sg-ab ,
711     Name-pl-ab ,
712     name-pl-ab ,
713 }
714 {
715     \keys_define:nn { zref-clever / typesetup }
716     {
717         #1 .default:V = \c_novaluel_tl ,
718         #1 .code:n =

```

```

719         {
720             \tl_if_novalue:nTF {##1}
721             {
722                 \prop_remove:cn
723                 { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
724                 {#1}
725             }
726             {
727                 \prop_put:cnn
728                 { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
729                 {#1} {##1}
730             }
731         } ,
732     }
733 }

```

## 5.2 \zcDeclareTranslations

\zcDeclareTranslations Provide \zcDeclareTranslations.

```

734 \NewDocumentCommand \zcDeclareTranslations { m m }
735 {
736     \tl_set:Nn \l__zrefclever_setup_language_tl {#1}
737     \tl_clear:N \l__zrefclever_setup_type_tl
738     \keys_set:nn { zref-clever / translations } {#2}
739 }

(End definition for \zcDeclareTranslations.)

740 \keys_define:nn { zref-clever / translations }
741 {
742     type .code:n =
743     {
744         \tl_if_empty:nTF {#1}
745         { \tl_clear:N \l__zrefclever_setup_type_tl }
746         {
747             \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
748             { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
749             \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
750         }
751     } ,
752 }

```

Not type-specific options.

```

753 \clist_map_inline:nn
754 {
755     tpairsep ,
756     tlistsep ,
757     tlastsep ,
758     notesep ,
759 }
760 {
761     \keys_define:nn { zref-clever / translations }
762     {
763         #1 .value_required:n = true ,
764         #1 .code:n =

```

```

765     {
766       \tl_if_empty:NTF \l__zrefclever_setup_type_tl
767       {
768         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
769         { zrefclever-default- #1 } {##1}
770       }
771       {
772         \msg_warning:nnn { zref-clever }
773         { option-not-type-specific } {#1}
774       }
775     } ,
776   }
777 }

```

Possibly type-specific options.

```

778 \clist_map_inline:nn
779 {
780   namesep ,
781   pairsep ,
782   listsep ,
783   lastsep ,
784   rangesep ,
785   refpre ,
786   refpos ,
787   refpre-in ,
788   refpos-in ,
789 }
790 {
791   \keys_define:nn { zref-clever / translations }
792   {
793     #1 .value_required:n = true ,
794     #1 .code:n =
795     {
796       \tl_if_empty:NTF \l__zrefclever_setup_type_tl
797       {
798         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
799         { zrefclever-default- #1 } {##1}
800       }
801       {
802         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
803         { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
804       }
805     } ,
806   }
807 }

```

Necessarily type-specific options.

```

808 \clist_map_inline:nn
809 {
810   Name-sg ,
811   name-sg ,
812   Name-pl ,
813   name-pl ,
814   Name-sg-ab ,
815   name-sg-ab ,

```

```

816     Name-pl-ab ,
817     name-pl-ab ,
818   }
819   {
820     \keys_define:nn { zref-clever / translations }
821     {
822       #1 .value_required:n = true ,
823       #1 .code:n =
824       {
825         \tl_if_empty:NTF \l__zrefclever_setup_type_tl
826         {
827           \msg_warning:nnn { zref-clever }
828             { option-only-type-specific } {#1}
829         }
830         {
831           \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
832             { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
833         }
834       } ,
835     }
836   }

```

## 6 \zcref

```

\zcref      \zcref{*}[\<options>]{\<labels>}
837 \NewDocumentCommand \zcref { s O { } m }
838 { \zref@wrapper@babel \__zrefclever_zcref:nnn {#3} {#1} {#2} }

```

*(End definition for \zcref.)*

```

\l__zrefclever_zcref_labels_seq
\l__zrefclever_link_star_bool
839 \seq_new:N \l__zrefclever_zcref_labels_seq
840 \bool_new:N \l__zrefclever_link_star_bool

```

*(End definition for \l\_\_zrefclever\_zcref\_labels\_seq and \l\_\_zrefclever\_link\_star\_bool.)*

`\__zrefclever_zcref:nnnn` An intermediate internal function, which does the actual heavy lifting, and places `{\<labels>}` as first argument, so that it can be protected by `\zref@wrapper@babel` in `\zcref`.

```

      \__zrefclever_zcref:nnnn {\<labels>} {\<*>} {\<options>}
841 \cs_new_protected:Npn \__zrefclever_zcref:nnn #1#2#3
842 {
843   \group_begin:
844     \keys_set:nn { zref-clever } {#3}
845     \seq_set_from_clist:Nn \l__zrefclever_zcref_labels_seq {#1}
846     \bool_set:Nn \l__zrefclever_link_star_bool {#2}
847     % Integration with 'zref-check'.
848     \bool_lazy_and:nnT
849       { \l__zrefclever_zrefcheck_available_bool }
850       { \l__zrefclever_zcref_with_check_bool }
851     { \zrefcheck_zcref_beg_label: }

```



```

852 \bool_lazy_or:nnT
853   { \l__zrefclever_typeset_sort_bool }
854   { \l__zrefclever_typeset_range_bool }
855   { \__zrefclever_sort_labels: }
856 \__zrefclever_typeset_refs:
857 % Typeset \texttt{note}.
858 \l__zrefclever_noteseq_tl
859 \l__zrefclever_zcref_note_tl
860 % Integration with 'zref-check'.
861 \bool_lazy_and:nnT
862   { \l__zrefclever_zrefcheck_available_bool }
863   { \l__zrefclever_zcref_with_check_bool }
864   {
865     \zrefcheck_zcref_end_label_maybe:
866     \zrefcheck_zcref_run_checks_on_labels:n
867     { \l__zrefclever_zcref_labels_seq }
868   }
869 \group_end:
870 }

```

(End definition for \\_\_zrefclever\_zcref:nnnn.)

## 7 \zcpageref

```

\zcpageref \zcpageref*<*>[<options>]{<labels>}
871 \NewDocumentCommand \zcpageref { s O { } m }
872 {
873   \IfBooleanTF {#1}
874     { \zcref*[#2, ref = page] {#3} }
875     { \zcref [ #2, ref = page] {#3} }
876 }

```

(End definition for \zcpageref.)

## 8 Sorting

```

877 \int_new:N \l__zrefclever_sort_prior_a_int
878 \int_new:N \l__zrefclever_sort_prior_b_int

```

\l\_\_zrefclever\_label\_a\_tl \l\_\_zrefclever\_label\_b\_tl  
Aux variables, for use in sorting and typesetting. I could probably let go some of them in favor of tmpa/tmpb, but they do improve code readability.

```

\l__zrefclever_label_type_a_tl \tl_new:N \l__zrefclever_label_a_tl
\l__zrefclever_label_type_b_tl \tl_new:N \l__zrefclever_label_b_tl
\l__zrefclever_label_enclcnt_a_tl \tl_new:N \l__zrefclever_label_type_a_tl
\l__zrefclever_label_enclcnt_b_tl \tl_new:N \l__zrefclever_label_type_b_tl
\l__zrefclever_label_enclval_a_tl \tl_new:N \l__zrefclever_label_enclcnt_a_tl
\l__zrefclever_label_enclval_b_tl \tl_new:N \l__zrefclever_label_enclcnt_b_tl
\l__zrefclever_label_enclval_a_tl \tl_new:N \l__zrefclever_label_enclval_a_tl
\l__zrefclever_label_enclval_b_tl \tl_new:N \l__zrefclever_label_enclval_b_tl

```

(End definition for \l\_\_zrefclever\_label\_a\_tl and others.)

`\l_zrefclever_label_types_seq` Stores the order in which reference types appear in the label list supplied by the user in `\zcref`. This order is required as a “last resort” sort criterion between the reference types, for use in `\__zrefclever_sort_default:nn`.

```
887 \seq_new:N \l__zrefclever_label_types_seq
```

(End definition for `\l__zrefclever_label_types_seq`.)

`\__zrefclever_sort_labels:` The main sorting function. It does not receive arguments, but it is expected to be run inside `\__zrefclever_zcref:nnnn` where a number of environment variables are to be set appropriately. In particular, `\l__zrefclever_zcref_labels_seq` should contain the labels received as argument to `\zcref`, and the function performs its task by sorting this variable.

```
888 \cs_new_protected:Npn \__zrefclever_sort_labels:
889 {
```

Store label types sequence.

```
890   \seq_clear:N \l__zrefclever_label_types_seq
891   \bool_if:NF \l__zrefclever_page_ref_bool
892   {
893     \seq_map_function:NN
894     \l__zrefclever_zcref_labels_seq \__zrefclever_label_type_put_new_right:n
895   }
```

Sort.

```
896   \seq_sort:Nn \l__zrefclever_zcref_labels_seq
897   {
898     \zref@ifrefundefined {##1}
899     {
900       \zref@ifrefundefined {##2}
901       {
902         % Neither label is defined.
903         \sort_return_same:
904       }
905       {
906         % The second label is defined, but the first isn't, leave the
907         % undefined first (to be more visible).
908         \sort_return_same:
909       }
910     }
911     {
912       \zref@ifrefundefined {##2}
913       {
914         % The first label is defined, but the second isn't, bring the
915         % second forward.
916         \sort_return_swapped:
917       }
918       {
919         % The interesting case: both labels are defined. The
920         % reference to the "default" property/counter or to the page
921         % are quite different from our perspective, they rely on
922         % different fields and even use different information for
923         % sorting, so we branch them here to specialized functions.
924         \bool_if:NTF \l__zrefclever_page_ref_bool
925         { \__zrefclever_sort_page:nn {##1} {##2} }
```

```

926             { \_zrefclever_sort_default:nn {##1} {##2} }
927         }
928     }
929 }
930 }

```

(End definition for \\_zrefclever\_sort\_labels:.)

\\_zrefclever\_label\_type\_put\_new\_right:n Auxiliary function used to store “new” label types (in order) as the sorting proceeds. It is expected to be run inside \\_zrefclever\_sort\_labels:, and stores new types in \l\\_zrefclever\_label\_types\_seq.

```

\_zrefclever_label_type_put_new_right:n {<label>}
931 \cs_new_protected:Npn \_zrefclever_label_type_put_new_right:n #1
932 {
933     \tl_set:Nx \l\_zrefclever_label_type_a_tl
934     { \zref@extractdefault {#1} { \c@type } { \c_empty_tl } }
935     \tl_if_empty:NF \l\_zrefclever_label_type_a_tl
936     {
937         \seq_if_in:NVF \l\_zrefclever_label_types_seq \l\_zrefclever_label_type_a_tl
938         {
939             \seq_put_right:NV
940             \l\_zrefclever_label_types_seq \l\_zrefclever_label_type_a_tl
941         }
942     }
943 }

```

(End definition for \\_zrefclever\_label\_type\_put\_new\_right:n.)

\l\\_zrefclever\_sort\_decided\_bool Auxiliary variable for \\_zrefclever\_sort\_default:nn, signals if the sorting between two labels has been decided or not.

```

944 \bool_new:N \l\_zrefclever_sort_decided_bool

```

(End definition for \l\\_zrefclever\_sort\_decided\_bool.)

\tl\_reverse\_items:V Variant not provided by the kernel.

```

945 \cs_generate_variant:Nn \tl_reverse_items:n { V }

```

(End definition for \tl\_reverse\_items:V.)

\\_zrefclever\_sort\_default:nn The heavy-lifting function for sorting of existing labels for “default” references (that is, a standard reference, not to “page”). This function is expected to be called within the sorting loop of \\_zrefclever\_sort\_labels: and receives the pair of labels being considered for a change of order or not. It should *always* “return” either \sort\_return\_same: or \sort\_return\_swapped:.

```

\_zrefclever_sort_default:nn {<label a>} {<label b>}
946 \cs_new_protected:Npn \_zrefclever_sort_default:nn #1#2
947 {
948     \tl_set:Nx \l\_zrefclever_label_type_a_tl
949     { \zref@extractdefault {#1} { \c@type } { \c_empty_tl } }
950     \tl_set:Nx \l\_zrefclever_label_type_b_tl
951     { \zref@extractdefault {#2} { \c@type } { \c_empty_tl } }
952 }

```

```

953 \bool_if:nTF
954 {
955   % The second label has a type, but the first doesn't, leave the
956   % undefined first (to be more visible).
957   \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
958   ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
959 }
960 { \sort_return_same: }
961 {
962   \bool_if:nTF
963   {
964     % The first label has a type, but the second doesn't, bring the
965     % second forward.
966     ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
967     \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
968   }
969   { \sort_return_swapped: }
970   {
971     \bool_if:nTF
972     {
973       % The interesting case: both labels have a type\dots{}
974       ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
975       ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
976     }
977     {
978       % Here we send this to a couple of auxiliary functions for no
979       % other reason than to keep this long function a little less
980       % unreadable.
981       \tl_if_eq:NNTF \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
982       {
983         % \dots{} and it's the same type.
984         \__zrefclever_sort_default_same_type:nn {#1} {#2}
985       }
986       {
987         % \dots{} and they are different types.
988         \__zrefclever_sort_default_different_types:nn {#1} {#2}
989       }
990     }
991   }
992   {
993     % Neither of the labels has a type. We can't do much of
994     % meaningful here, but if it's the same counter, compare it.
995     \exp_args:Nxx \tl_if_eq:nnTF
996     { \zref@extractdefault {#1} { counter } { } }
997     { \zref@extractdefault {#2} { counter } { } }
998     {
999       \int_compare:nNnTF
1000       { \zref@extractdefault {#1} { zc@cntval } {-1} }
1001       >
1002       { \zref@extractdefault {#2} { zc@cntval } {-1} }
1003       { \sort_return_swapped: }
1004       { \sort_return_same: }
1005     }
1006   }
1007 }

```

```

1007     }
1008   }
1009 }

```

(End definition for \\_zrefclever\_sort\_default:nn.)

\\_zrefclever\_sort\_default\_same\_type:nn

```

1010 \cs_new_protected:Npn \_zrefclever_sort_default_same_type:nn #1#2
1011 {
1012   \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1013     { \zref@extractdefault {#1} { zc@enclcnt } { \c_empty_tl } }
1014   \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1015     { \tl_reverse_items:V \l__zrefclever_label_enclcnt_a_tl }
1016   \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1017     { \zref@extractdefault {#2} { zc@enclcnt } { \c_empty_tl } }
1018   \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1019     { \tl_reverse_items:V \l__zrefclever_label_enclcnt_b_tl }
1020   \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1021     { \zref@extractdefault {#1} { zc@enclval } { \c_empty_tl } }
1022   \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1023     { \tl_reverse_items:V \l__zrefclever_label_enclval_a_tl }
1024   \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1025     { \zref@extractdefault {#2} { zc@enclval } { \c_empty_tl } }
1026   \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1027     { \tl_reverse_items:V \l__zrefclever_label_enclval_b_tl }
1028
1029   \bool_set_false:N \l__zrefclever_sort_decided_bool
1030   % CHECK should I replace the tmp variables here?
1031   \tl_clear:N \l_tmpa_tl
1032   \tl_clear:N \l_tmpb_tl
1033   \bool_until_do:Nn \l__zrefclever_sort_decided_bool
1034   {
1035     \tl_set:Nx \l_tmpa_tl
1036       { \tl_head:N \l__zrefclever_label_enclcnt_a_tl }
1037     \tl_set:Nx \l_tmpb_tl
1038       { \tl_head:N \l__zrefclever_label_enclcnt_b_tl }
1039
1040     \bool_if:nTF
1041     {
1042       % Both are empty, meaning: neither labels have any (further)
1043       % ‘‘enclosing counters’’ (left).
1044       \tl_if_empty_p:V \l_tmpa_tl &&
1045       \tl_if_empty_p:V \l_tmpb_tl
1046     }
1047     {
1048       \exp_args:Nxx \tl_if_eq:nnTF
1049       { \zref@extractdefault {#1} { counter } { } }
1050       { \zref@extractdefault {#2} { counter } { } }
1051       {
1052         \bool_set_true:N \l__zrefclever_sort_decided_bool
1053         \int_compare:nNnTF
1054         { \zref@extractdefault {#1} { zc@cntval } {-1} }
1055         >
1056         { \zref@extractdefault {#2} { zc@cntval } {-1} }

```

```

1057         { \sort_return_swapped: }
1058         { \sort_return_same: }
1059     }
1060     {
1061         \msg_warning:nnnn { zref-clever }
1062         { counters-not-nested } {#1} {#2}
1063         \bool_set_true:N \l__zrefclever_sort_decided_bool
1064         \sort_return_same:
1065     }
1066 }
1067 {
1068     \bool_if:nTF
1069     {
1070         % 'a' is empty (and 'b' is not), meaning: 'b' is (possibly)
1071         % nested in 'a'.
1072         \tl_if_empty_p:V \l_tmpa_tl
1073     }
1074     {
1075         \tl_set:Nx \l_tmpa_tl
1076         { {\zref@extractdefault {#1} { counter } { }} }
1077         \exp_args:NNx \tl_if_in:NnTF
1078         \l__zrefclever_label_enclcnt_b_tl { \l_tmpa_tl }
1079         {
1080             \bool_set_true:N \l__zrefclever_sort_decided_bool
1081             \sort_return_same:
1082         }
1083         {
1084             \msg_warning:nnnn { zref-clever }
1085             { counters-not-nested } {#1} {#2}
1086             \bool_set_true:N \l__zrefclever_sort_decided_bool
1087             \sort_return_same:
1088         }
1089     }
1090 }
1091 {
1092     \bool_if:nTF
1093     {
1094         % 'b' is empty (and 'a' is not), meaning: 'a' is
1095         % (possibly) nested in 'b'.
1096         \tl_if_empty_p:V \l_tmpb_tl
1097     }
1098     {
1099         \tl_set:Nx \l_tmpb_tl
1100         { {\zref@extractdefault {#2} { counter } { }} }
1101         \exp_args:NNx \tl_if_in:NnTF
1102         \l__zrefclever_label_enclcnt_a_tl { \l_tmpb_tl }
1103         {
1104             \bool_set_true:N \l__zrefclever_sort_decided_bool
1105             \sort_return_swapped:
1106         }
1107         {
1108             \msg_warning:nnnn { zref-clever }
1109             { counters-not-nested } {#1} {#2}
1110             \bool_set_true:N \l__zrefclever_sort_decided_bool
1111             \sort_return_same:

```

```

1111     }
1112   }
1113   {
1114     % Neither is empty, meaning: we can (possibly) compare the
1115     % values of the current enclosing counter in the loop, if
1116     % they are equal, we are still in the loop, if they are
1117     % not, a sorting decision can be made directly.
1118     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
1119     {
1120       \int_compare:nNnTF
1121         { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1122         =
1123         { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1124         {
1125           \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1126             { \tl_tail:N \l__zrefclever_label_enclcnt_a_tl }
1127           \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1128             { \tl_tail:N \l__zrefclever_label_enclcnt_b_tl }
1129           \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1130             { \tl_tail:N \l__zrefclever_label_enclval_a_tl }
1131           \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1132             { \tl_tail:N \l__zrefclever_label_enclval_b_tl }
1133         }
1134         {
1135           \bool_set_true:N \l__zrefclever_sort_decided_bool
1136           \int_compare:nNnTF
1137             { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1138             >
1139             { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1140             { \sort_return_swapped: }
1141             { \sort_return_same: }
1142         }
1143       }
1144     {
1145       \msg_warning:nnnn { zref-clever }
1146       { counters-not-nested } {#1} {#2}
1147       \bool_set_true:N \l__zrefclever_sort_decided_bool
1148       \sort_return_same:
1149     }
1150   }
1151 }
1152 }
1153 }
1154 }

```

(End definition for \\_\_zrefclever\_sort\_default\_same\_type:nn.)

\_zrefclever\_sort\_default\_different\_types:nn

```

1155 \cs_new_protected:Npn \__zrefclever_sort_default_different_types:nn #1#2
1156 {
1157   \int_zero:N \l__zrefclever_sort_prior_a_int
1158   \int_zero:N \l__zrefclever_sort_prior_b_int
1159   % \cs{l__zrefclever_typesort_seq} was stored in reverse sequence, and we compute
1160   % the sort priorities in the negative range, so that we can implicitly

```

```

1161 % rely on '0' being the 'last value'.
1162 \seq_map_indexed_inline:Nn \l__zrefclever_typesort_seq
1163 {
1164   \tl_if_eq:nnTF {##2} {{othertypes}}
1165   {
1166     \int_compare:nNnT { \l__zrefclever_sort_prior_a_int } = { 0 }
1167     { \int_set:Nn \l__zrefclever_sort_prior_a_int { - ##1 } }
1168     \int_compare:nNnT { \l__zrefclever_sort_prior_b_int } = { 0 }
1169     { \int_set:Nn \l__zrefclever_sort_prior_b_int { - ##1 } }
1170   }
1171   {
1172     \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##2}
1173     { \int_set:Nn \l__zrefclever_sort_prior_a_int { - ##1 } }
1174     {
1175       \tl_if_eq:NnT \l__zrefclever_label_type_b_tl {##2}
1176       { \int_set:Nn \l__zrefclever_sort_prior_b_int { - ##1 } }
1177     }
1178   }
1179 }
1180 \bool_if:nTF
1181 {
1182   \int_compare_p:nNn
1183   { \l__zrefclever_sort_prior_a_int } <
1184   { \l__zrefclever_sort_prior_b_int }
1185 }
1186 { \sort_return_same: }
1187 {
1188   \bool_if:nTF
1189   {
1190     \int_compare_p:nNn
1191     { \l__zrefclever_sort_prior_a_int } >
1192     { \l__zrefclever_sort_prior_b_int }
1193   }
1194   { \sort_return_swapped: }
1195   {
1196     % Sort priorities are equal for different types: the type that
1197     % occurs first in \meta{labels}, as given by the user, is kept (or
1198     % brought) forward.
1199     \seq_map_inline:Nn \l__zrefclever_label_types_seq
1200     {
1201       \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##1}
1202       { \seq_map_break:n { \sort_return_same: } }
1203       {
1204         \tl_if_eq:NnT \l__zrefclever_label_type_b_tl {##1}
1205         { \seq_map_break:n { \sort_return_swapped: } }
1206       }
1207     }
1208   }
1209 }
1210 }

```

(End definition for `\__zrefclever_sort_default_different_types:nn`.)

`\__zrefclever_sort_page:nn` The sorting function for sorting of existing labels for references to “page”. This function is expected to be called within the sorting loop of `\__zrefclever_sort_labels:` and



receives the pair of labels being considered for a change of order or not. It should *always* “return” either `\sort_return_same:` or `\sort_return_swapped:`. Compared to the sorting of default labels, this is a piece of cake (thanks to `abspage`).

```

    \__zrefclever_sort_page:nn {\label a} {\label b}

1211 \cs_new_protected:Npn \__zrefclever_sort_page:nn #1#2
1212 {
1213   \int_compare:nNnTF
1214     { \zref@extractdefault {#1} { abspage } {-1} }
1215     >
1216     { \zref@extractdefault {#2} { abspage } {-1} }
1217     { \sort_return_swapped: }
1218     { \sort_return_same:   }
1219 }

```

(End definition for `\__zrefclever_sort_page:nn`.)

## 9 Typesetting

About possible alternatives to signal compression inhibition for individual labels, see <https://tex.stackexchange.com/q/611370> (thanks Enrico Gregorio, Phelype Oleinik, and Steven B. Segletes). Yet another alternative would be to receive an optional argument with the label(s) not to be compressed. This would be a repetition, but would keep the syntax “clean”. All in all, and rethinking this here, probably the best is simply to not allow individual inhibition of compression. We can already control compression of each individual call of `\zcref` with existing options, this should be enough. I don’t think the small extra flexibility this would grant is worth the syntax disruption it entails. Anyway, I have kept a “handle” to deal with this in case the need arises, in the form of `\l__zrefclever_range_inhibit_next_bool`, which is currently no-op, but is in place.

### Typesetting variables

```

\l_zrefclever_typeset_last_bool
\l_zrefclever_last_of_type_bool

```

Auxiliary variables for `\__zrefclever_typeset_refs:`. `\l__zrefclever_typeset_last_bool` signals if the label list is over so that we can leave the loop. `\l__zrefclever_last_of_type_bool` signals if we are processing the last label of the current reference type.

```

1220 \bool_new:N \l__zrefclever_typeset_last_bool
1221 \bool_new:N \l__zrefclever_last_of_type_bool

```

(End definition for `\l__zrefclever_typeset_last_bool` and `\l__zrefclever_last_of_type_bool`.)

```

\l_zrefclever_typeset_labels_seq
\l_zrefclever_typeset_queue_prev_tl
\l_zrefclever_typeset_queue_curr_tl
\l_zrefclever_type_first_label_tl
\l_zrefclever_type_first_label_type_tl

```

Auxiliary variables for `\__zrefclever_typeset_refs:`. They store, respectively the “previous” and the “current” reference type information while they are being processed, since we cannot typeset them directly, given we can only know certain things when the (next) type list is over. The “queue” stores all references but the first of the type, and they are stored ready to be typeset. The “first\_label” stores the *label* of the first reference for the type, because the name can only be determined at the end, and its (potential) hyperlink must be handled at that point.

```

1222 \seq_new:N \l__zrefclever_typeset_labels_seq
1223 \tl_new:N \l__zrefclever_typeset_queue_prev_tl
1224 \tl_new:N \l__zrefclever_typeset_queue_curr_tl

```

```

1225 \tl_new:N \l__zrefclever_type_first_label_tl
1226 \tl_new:N \l__zrefclever_type_first_label_type_tl

```

(End definition for \l\_\_zrefclever\_typeset\_labels\_seq and others.)

\l\_\_zrefclever\_label\_count\_int    Main counters for \\_\_zrefclever\_typeset\_refs:. They track the state of the parsing of the labels list. \l\_\_zrefclever\_label\_count\_int is stepped for every reference/label in the list, and reset at the start of a new type. \l\_\_zrefclever\_type\_count\_int is stepped at every reference type change.

```

1227 \int_new:N \l__zrefclever_label_count_int
1228 \int_new:N \l__zrefclever_type_count_int

```

(End definition for \l\_\_zrefclever\_label\_count\_int and \l\_\_zrefclever\_type\_count\_int.)

\l\_\_zrefclever\_range\_count\_int    Range related auxiliary variables for \\_\_zrefclever\_typeset\_refs:. \l\_\_zrefclever\_range\_count\_int counts how many references/labels are in the current ongoing range. \l\_\_zrefclever\_range\_same\_count\_int counts how many of the references in the current ongoing range are repeated ones. \l\_\_zrefclever\_range\_beg\_label\_tl stores the label of the reference that starts a range. \l\_\_zrefclever\_next\_maybe\_range\_bool signals whether the next element is in sequence to the current one. \l\_\_zrefclever\_next\_is\_same\_bool signals whether the next element repeats the current one. \l\_\_zrefclever\_range\_inhibit\_next\_bool allows to control/track compression inhibition of the next label.

```

1229 \int_new:N \l__zrefclever_range_count_int
1230 \int_new:N \l__zrefclever_range_same_count_int
1231 \tl_new:N \l__zrefclever_range_beg_label_tl
1232 \bool_new:N \l__zrefclever_next_maybe_range_bool
1233 \bool_new:N \l__zrefclever_next_is_same_bool
1234 \bool_new:N \l__zrefclever_range_inhibit_next_bool

```

(End definition for \l\_\_zrefclever\_range\_count\_int and others.)

Aux variables for \\_\_zrefclever\_typeset\_refs:. Store separators and refpre/pos options.

```

1235 \tl_new:N \l__zrefclever_namefont_tl
1236 \tl_new:N \l__zrefclever_reffont_out_tl
1237 \tl_new:N \l__zrefclever_reffont_in_tl
1238
1239 \tl_new:N \l__zrefclever_namesep_tl
1240 \tl_new:N \l__zrefclever_rangesep_tl
1241 \tl_new:N \l__zrefclever_pairsep_tl
1242 \tl_new:N \l__zrefclever_listsep_tl
1243 \tl_new:N \l__zrefclever_lastsep_tl
1244 % ‘t’ for ‘type’
1245 \tl_new:N \l__zrefclever_tpairsep_tl
1246 \tl_new:N \l__zrefclever_tlistsep_tl
1247 \tl_new:N \l__zrefclever_tlastsep_tl
1248 \tl_new:N \l__zrefclever_notesep_tl
1249 \tl_new:N \l__zrefclever_refpre_out_tl
1250 \tl_new:N \l__zrefclever_refpos_out_tl
1251 \tl_new:N \l__zrefclever_refpre_in_tl
1252 \tl_new:N \l__zrefclever_refpos_in_tl

```

(End definition for .)

```

\l__zrefclever_type_name_tl Auxiliary variables for \__zrefclever_get_ref_first: and \__zrefclever_type_
\l__zrefclever_name_in_link_bool name_setup:.
\l__zrefclever_name_format_tl
\l__zrefclever_name_format_fallback_tl
1253 \tl_new:N \l__zrefclever_type_name_tl
1254 \bool_new:N \l__zrefclever_name_in_link_bool
1255 \tl_new:N \l__zrefclever_name_format_tl
1256 \tl_new:N \l__zrefclever_name_format_fallback_tl

(End definition for \l__zrefclever_type_name_tl and others.)

```

## Main typesetting functions

```

\__zrefclever_typeset_refs: Main typesetting function for \zcref.
1257 \cs_new_protected:Npn \__zrefclever_typeset_refs:
1258 {
1259   \seq_set_eq:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_zcref_labels_seq
1260   \tl_clear:N \l__zrefclever_typeset_queue_prev_tl
1261   \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1262   \tl_clear:N \l__zrefclever_type_first_label_tl
1263   \tl_clear:N \l__zrefclever_type_first_label_type_tl
1264   \tl_clear:N \l__zrefclever_range_beg_label_tl
1265   \int_zero:N \l__zrefclever_label_count_int
1266   \int_zero:N \l__zrefclever_type_count_int
1267   \int_zero:N \l__zrefclever_range_count_int
1268   \int_zero:N \l__zrefclever_range_same_count_int
1269
1270   % Get not-type-specific separators and refpre/pos options.
1271   \__zrefclever_get_option_with_transl:nN {tpairsep} \l__zrefclever_tpairsep_tl
1272   \__zrefclever_get_option_with_transl:nN {tlistsep} \l__zrefclever_tlistsep_tl
1273   \__zrefclever_get_option_with_transl:nN {tlastsep} \l__zrefclever_tlastsep_tl
1274   \__zrefclever_get_option_with_transl:nN {notesep} \l__zrefclever_notesep_tl
1275
1276   % Set the font option for this zcref call.
1277   \l__zrefclever_ref_typeset_font_tl
1278
1279   % Loop over the label list in sequence.
1280   \bool_set_false:N \l__zrefclever_typeset_last_bool
1281   \bool_until_do:Nn \l__zrefclever_typeset_last_bool
1282   {
1283     \seq_pop_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_a_tl
1284     \seq_if_empty:NTF \l__zrefclever_typeset_labels_seq
1285     {
1286       \tl_clear:N \l__zrefclever_label_b_tl
1287       \bool_set_true:N \l__zrefclever_typeset_last_bool
1288     }
1289     { \seq_get_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_b_tl }
1290
1291     \bool_if:NTF \l__zrefclever_page_ref_bool
1292     {
1293       \tl_set:Nn \l__zrefclever_label_type_a_tl { page }
1294       \tl_set:Nn \l__zrefclever_label_type_b_tl { page }
1295     }
1296     {
1297       \tl_set:Nx \l__zrefclever_label_type_a_tl
1298       {

```

```

1299         \zref@extractdefault
1300         { \l__zrefclever_label_a_tl } { zc@type } { \c_empty_tl }
1301     }
1302     \tl_set:Nx \l__zrefclever_label_type_b_tl
1303     {
1304         \zref@extractdefault
1305         { \l__zrefclever_label_b_tl } { zc@type } { \c_empty_tl }
1306     }
1307 }
1308
1309 % First, we establish whether the ‘current label’ (i.e. ‘a’) is the
1310 % last one of its type. This can happen because the ‘next label’
1311 % (i.e. ‘b’) is of a different type (or different definition status),
1312 % or because we are at the end of the list.
1313 \bool_if:NTF \l__zrefclever_typeset_last_bool
1314 { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1315 {
1316     \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1317     {
1318         \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1319         { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1320         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1321     }
1322     {
1323         \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1324         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1325         {
1326             % Neither is undefined, we must check the types.
1327             \bool_if:nTF
1328             % Both empty: same ‘type’.
1329             {
1330                 \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1331                 \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1332             }
1333             { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1334             {
1335                 \bool_if:nTF
1336                 % Neither empty: compare types.
1337                 {
1338                     ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1339                     ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1340                 }
1341                 {
1342                     \tl_if_eq:NNTF
1343                     \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
1344                     { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1345                     { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1346                 }
1347                 % One empty, the other not: different ‘types’.
1348                 { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1349             }
1350         }
1351     }
1352 }

```

```

1353
1354 % Handle warnings in case of reference or type undefined.
1355 \zref@refused { \l__zrefclever_label_a_tl }
1356 \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1357 {}
1358 {
1359   \tl_if_empty:NT \l__zrefclever_label_type_a_tl
1360   {
1361     \msg_warning:nxx { zref-clever } { missing-type }
1362     { \l__zrefclever_label_a_tl }
1363   }
1364 }
1365
1366 % Get type-specific separators, refpre/pos and font options, once per
1367 % type.
1368 \int_compare:nNnT { \l__zrefclever_label_count_int } = { 0 }
1369 {
1370   \__zrefclever_get_option_plain:nN {namefont}      \l__zrefclever_namefont_tl
1371   \__zrefclever_get_option_plain:nN {reffont}      \l__zrefclever_reffont_out_tl
1372   \__zrefclever_get_option_plain:nN {reffont-in}   \l__zrefclever_reffont_in_tl
1373   \__zrefclever_get_option_with_transl:nN {namesep} \l__zrefclever_namesep_tl
1374   \__zrefclever_get_option_with_transl:nN {rangesep} \l__zrefclever_rangesep_tl
1375   \__zrefclever_get_option_with_transl:nN {pairsep} \l__zrefclever_pairsep_tl
1376   \__zrefclever_get_option_with_transl:nN {listsep} \l__zrefclever_listsep_tl
1377   \__zrefclever_get_option_with_transl:nN {lastsep} \l__zrefclever_lastsep_tl
1378   \__zrefclever_get_option_with_transl:nN {refpre}  \l__zrefclever_refpre_out_tl
1379   \__zrefclever_get_option_with_transl:nN {refpos}  \l__zrefclever_refpos_out_tl
1380   \__zrefclever_get_option_with_transl:nN {refpre-in} \l__zrefclever_refpre_in_tl
1381   \__zrefclever_get_option_with_transl:nN {refpos-in} \l__zrefclever_refpos_in_tl
1382 }
1383
1384 % Here we send this to a couple of auxiliary functions for no other
1385 % reason than to keep this long function a little less unreadable.
1386 \bool_if:NTF \l__zrefclever_last_of_type_bool
1387 {
1388   % There exists no next label of the same type as the current.
1389   \__zrefclever_typeset_refs_aux_last_of_type:
1390 }
1391 {
1392   % There exists a next label of the same type as the current.
1393   \__zrefclever_typeset_refs_aux_not_last_of_type:
1394 }
1395 }
1396 }

```

(End definition for \\_\_zrefclever\_typeset\_refs:.)

\\_\_zrefclever\_typeset\_refs\_aux\_last\_of\_type: Handles typesetting of when the current label is the last of its type.

```

1397 \cs_new_protected:Npn \__zrefclever_typeset_refs_aux_last_of_type:
1398 {
1399   % Process the current label to the current queue.
1400   \int_case:nnF { \l__zrefclever_label_count_int }
1401   {
1402     % It is the last label of its type, but also the first one, and that's

```

```

1403 % what matters here: just store it.
1404 { 0 }
1405 {
1406   \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1407   \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1408 }
1409
1410 % The last is the second: we have a pair (if not repeated).
1411 { 1 }
1412 {
1413   \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1414   {
1415     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1416     {
1417       \exp_not:V \l__zrefclever_pairsep_tl
1418       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1419     }
1420   }
1421 }
1422 }
1423 % If neither the first, nor the second: we have the last label
1424 % on the current type list (if not repeated).
1425 {
1426   \int_case:nnF { \l__zrefclever_range_count_int }
1427   {
1428     % There was no range going on.
1429     {0}
1430     {
1431       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1432       {
1433         \exp_not:V \l__zrefclever_lastsep_tl
1434         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1435       }
1436     }
1437     % Last in the range is also the second in it.
1438     {1}
1439     {
1440       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1441       {
1442         % We know 'range_beg_label' is not empty, since this is the
1443         % second element in the range, but the third or more in the
1444         % type list.
1445         \exp_not:V \l__zrefclever_listsep_tl
1446         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1447         \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1448         {
1449           \exp_not:V \l__zrefclever_lastsep_tl
1450           \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1451         }
1452       }
1453     }
1454   }
1455   % Last in the range is third or more in it.
1456   {

```

```

1457 \int_case:nnF
1458 { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1459 {
1460   % Repetition, not a range.
1461   {0}
1462   {
1463     % If 'range_beg_label' is empty, it means it was also the
1464     % first of the type, and hence was already handled.
1465     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1466     {
1467       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1468       {
1469         \exp_not:V \l__zrefclever_lastsep_tl
1470         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1471       }
1472     }
1473   }
1474   % A 'range', but with no skipped value, treat as list.
1475   {1}
1476   {
1477     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1478     {
1479       % Ditto.
1480       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1481       {
1482         \exp_not:V \l__zrefclever_listsep_tl
1483         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1484       }
1485       \exp_not:V \l__zrefclever_lastsep_tl
1486       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1487     }
1488   }
1489 }
1490 {
1491   % An actual range.
1492   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1493   {
1494     % Ditto.
1495     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1496     {
1497       \exp_not:V \l__zrefclever_lastsep_tl
1498       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1499     }
1500     \exp_not:V \l__zrefclever_rangesep_tl
1501     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1502   }
1503 }
1504 }
1505 }
1506
1507 % Handle 'range' option. The idea is simple: if the queue is not empty,
1508 % we replace it with the end of the range (or pair). We can still
1509 % retrieve the end of the range from \cs{l__zrefclever_label_a_tl} since we know to
1510 % be processing the last label of its type at this point.

```

```

1511 \bool_if:NT \l__zrefclever_typeset_range_bool
1512 {
1513   \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1514   {
1515     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1516     { }
1517     {
1518       \msg_warning:nxx { zref-clever } { single-element-range }
1519       { \l__zrefclever_type_first_label_type_tl }
1520     }
1521   }
1522   {
1523     \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1524     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1525     { }
1526     {
1527       \__zrefclever_labels_in_sequence:nn
1528       { \l__zrefclever_type_first_label_tl } { \l__zrefclever_label_a_tl }
1529     }
1530     \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1531     {
1532       \bool_if:NTF \l__zrefclever_next_maybe_range_bool
1533       { \exp_not:V \l__zrefclever_pairsep_tl }
1534       { \exp_not:V \l__zrefclever_rangesep_tl }
1535       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1536     }
1537   }
1538 }
1539
1540 % Now that the type is finished, we can add the name and the first ref to
1541 % the queue. Or, if ‘‘typset’’ option is not ‘‘both’’, handle it here
1542 % too.
1543 \__zrefclever_type_name_setup:
1544 \bool_if:nTF
1545 { \l__zrefclever_typeset_ref_bool && \l__zrefclever_typeset_name_bool }
1546 {
1547   \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1548   { \__zrefclever_get_ref_first: }
1549 }
1550 {
1551   \bool_if:nTF
1552   { \l__zrefclever_typeset_ref_bool }
1553   {
1554     \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1555     { \__zrefclever_get_ref:V \l__zrefclever_type_first_label_tl }
1556   }
1557   {
1558     \bool_if:nTF
1559     { \l__zrefclever_typeset_name_bool }
1560     {
1561       \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1562       {
1563         \bool_if:NTF \l__zrefclever_name_in_link_bool
1564         {

```



```

1565 \exp_not:N \group_begin:
1566 \exp_not:V \l__zrefclever_namefont_tl
1567 % It's two '@s', but escaped for DocStrip.
1568 \exp_not:N \hyper@@link
1569 {
1570   \zref@ifrefcontainsprop
1571     { \l__zrefclever_type_first_label_tl } { urluse }
1572     {
1573       \zref@extractdefault
1574         { \l__zrefclever_type_first_label_tl }
1575         { urluse } {}
1576     }
1577     {
1578       \zref@extractdefault
1579         { \l__zrefclever_type_first_label_tl }
1580         { url } {}
1581     }
1582   }
1583   {
1584     \zref@extractdefault
1585       { \l__zrefclever_type_first_label_tl } { anchor } {}
1586   }
1587   { \exp_not:V \l__zrefclever_type_name_tl }
1588 \exp_not:N \group_end:
1589 }
1590 {
1591   \exp_not:N \group_begin:
1592   \exp_not:V \l__zrefclever_namefont_tl
1593   \exp_not:V \l__zrefclever_type_name_tl
1594   \exp_not:N \group_end:
1595 }
1596 }
1597 }
1598 {
1599   % This case would correspond to "typeset=none" but should not
1600   % happen, given the options are set up to typeset at least one
1601   % of "ref" or "name", but a sensible fallback, equal to the
1602   % behavior of "both".
1603   \tl_put_left:Nx
1604     \l__zrefclever_typeset_queue_curr_tl { \__zrefclever_get_ref_first: }
1605   }
1606 }
1607 }
1608
1609 % Typeset the previous type, if there is one.
1610 \int_compare:nNnT { \l__zrefclever_type_count_int } > { 0 }
1611 {
1612   \int_compare:nNnT { \l__zrefclever_type_count_int } > { 1 }
1613     { \l__zrefclever_tlistsep_tl }
1614   \l__zrefclever_typeset_queue_prev_tl
1615 }
1616
1617 % Wrap up loop, or prepare for next iteration.
1618 \bool_if:NTF \l__zrefclever_typeset_last_bool

```

```

1619 {
1620   % We are finishing, typeset the current queue.
1621   \int_case:nnF { \l__zrefclever_type_count_int }
1622   {
1623     % Single type.
1624     { 0 }
1625     { \l__zrefclever_typeset_queue_curr_tl }
1626     % Pair of types.
1627     { 1 }
1628     {
1629       \l__zrefclever_tpairsep_tl
1630       \l__zrefclever_typeset_queue_curr_tl
1631     }
1632   }
1633   {
1634     % Last in list of types.
1635     \l__zrefclever_tlastsep_tl
1636     \l__zrefclever_typeset_queue_curr_tl
1637   }
1638 }
1639 {
1640   % There are further labels, set variables for next iteration.
1641   \tl_set_eq:NN
1642     \l__zrefclever_typeset_queue_prev_tl \l__zrefclever_typeset_queue_curr_tl
1643   \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1644   \tl_clear:N \l__zrefclever_type_first_label_tl
1645   \tl_clear:N \l__zrefclever_type_first_label_type_tl
1646   \tl_clear:N \l__zrefclever_range_beg_label_tl
1647   \int_zero:N \l__zrefclever_label_count_int
1648   \int_incr:N \l__zrefclever_type_count_int
1649   \int_zero:N \l__zrefclever_range_count_int
1650   \int_zero:N \l__zrefclever_range_same_count_int
1651 }
1652 }

```

(End definition for \\_\_zrefclever\_typeset\_refs\_aux\_last\_of\_type:.)

\zrefclever\_typeset\_refs\_aux\_not\_last\_of\_type: Handles typesetting of when the current label is not the last of its type.

```

1653 \cs_new_protected:Npn \__zrefclever_typeset_refs_aux_not_last_of_type:
1654 {
1655   % Signal if next label may form a range with the current one (of
1656   % course, only considered if compression is enabled in the first
1657   % place).
1658   \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1659   \bool_set_false:N \l__zrefclever_next_is_same_bool
1660   \bool_lazy_and:nnT
1661     { \l__zrefclever_typeset_compress_bool }
1662     % Currently no-op, but kept as ‘‘handle’’ to inhibit compression of
1663     % individual labels.
1664     { ! \l__zrefclever_range_inhibit_next_bool }
1665   {
1666     \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1667     { }
1668     {

```

```

1669         \l__zrefclever_labels_in_sequence:nn
1670         { \l__zrefclever_label_a_tl } { \l__zrefclever_label_b_tl }
1671     }
1672 }
1673
1674 % Process the current label to the current queue.
1675 \int_compare:nNnTF { \l__zrefclever_label_count_int } = { 0 }
1676 {
1677     % Current label is the first of its type (also not the last, but it
1678     % doesn't matter here): just store the label.
1679     \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1680     \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1681
1682     % If the next label may be part of a range, we set 'range_beg_label'
1683     % to 'empty' (we deal with it as the 'first', and must do it
1684     % there, to handle hyperlinking), but also step the range counters.
1685     \bool_if:NT \l__zrefclever_next_maybe_range_bool
1686     {
1687         \tl_clear:N \l__zrefclever_range_beg_label_tl
1688         \int_incr:N \l__zrefclever_range_count_int
1689         \bool_if:NT \l__zrefclever_next_is_same_bool
1690         { \int_incr:N \l__zrefclever_range_same_count_int }
1691     }
1692 }
1693 {
1694     % Current label is neither the first (nor the last) of its
1695     % type.
1696     \bool_if:NTF \l__zrefclever_next_maybe_range_bool
1697     {
1698         % Starting, or continuing a range.
1699         \int_compare:nNnTF
1700         { \l__zrefclever_range_count_int } = {0}
1701         {
1702             % There was no range going, we are starting one.
1703             \tl_set:NV \l__zrefclever_range_beg_label_tl \l__zrefclever_label_a_tl
1704             \int_incr:N \l__zrefclever_range_count_int
1705             \bool_if:NT \l__zrefclever_next_is_same_bool
1706             { \int_incr:N \l__zrefclever_range_same_count_int }
1707         }
1708         {
1709             % Second or more in the range, but not the last.
1710             \int_incr:N \l__zrefclever_range_count_int
1711             \bool_if:NT \l__zrefclever_next_is_same_bool
1712             { \int_incr:N \l__zrefclever_range_same_count_int }
1713         }
1714     }
1715 }
1716 % Next element is not in sequence, meaning: there was no range, or
1717 % we are closing one.
1718 \int_case:nnF { \l__zrefclever_range_count_int }
1719 {
1720     % There was no range going on.
1721     {0}
1722     {

```

```

1723 \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1724 {
1725   \exp_not:V \l__zrefclever_listsep_tl
1726   \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1727 }
1728 }
1729 % Last is second in the range: if 'range_same_count' is also
1730 % '1', it's a repetition (drop it), otherwise, it's a 'pair
1731 % within a list'', treat as list.
1732 {1}
1733 {
1734   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1735   {
1736     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1737     {
1738       \exp_not:V \l__zrefclever_listsep_tl
1739       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1740     }
1741     \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1742     {
1743       \exp_not:V \l__zrefclever_listsep_tl
1744       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1745     }
1746   }
1747 }
1748 }
1749 {
1750 % Last is third or more in the range: if 'range_count' and
1751 % 'range_same_count' are the same, its a repetition (drop it),
1752 % if they differ by '1', its a list, if they differ by more,
1753 % it is a real range.
1754 \int_case:nnF
1755 { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1756 {
1757   {0}
1758   {
1759     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1760     {
1761       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1762       {
1763         \exp_not:V \l__zrefclever_listsep_tl
1764         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1765       }
1766     }
1767   }
1768   {1}
1769   {
1770     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1771     {
1772       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1773       {
1774         \exp_not:V \l__zrefclever_listsep_tl
1775         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1776       }

```

```

1777         \exp_not:V \l__zrefclever_listsep_tl
1778         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1779     }
1780 }
1781 }
1782 {
1783     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1784     {
1785         \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1786         {
1787             \exp_not:V \l__zrefclever_listsep_tl
1788             \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1789         }
1790         \exp_not:V \l__zrefclever_rangesep_tl
1791         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1792     }
1793 }
1794 }
1795 % Reset counters.
1796 \int_zero:N \l__zrefclever_range_count_int
1797 \int_zero:N \l__zrefclever_range_same_count_int
1798 }
1799 }
1800 % Step label counter for next iteration.
1801 \int_incr:N \l__zrefclever_label_count_int
1802 }

```

(End definition for \\_\_zrefclever\_typeset\_refs\_aux\_not\_last\_of\_type:.)

## Aux typesetting functions

`\__zrefclever_get_ref:n` Auxiliary function to `\__zrefclever_typeset_refs:.` Handles a complete “ref-block”, including “pre” and “pos” elements, and *hyperlinking*. It does not handle the reference type “name”, for that use `\__zrefclever_get_ref_first:.` It should get the reference with `\zref@extractdefault` as usual but, if the reference is not available, should put `\zref@default` on the stream protected, so that it can be accumulated in the queue. `\hyperlink` must also be protected from expansion for the same reason.

```

1803 \cs_new:Npn \__zrefclever_get_ref:n #1
1804 {
1805     \zref@ifrefcontainsprop {#1} { \l__zrefclever_ref_property_tl }
1806     {
1807         \bool_if:nTF
1808         { \l__zrefclever_use_hyperref_bool && ! \l__zrefclever_link_star_bool }
1809         {
1810             \exp_not:N \group_begin:
1811             \exp_not:V \l__zrefclever_reffont_out_tl
1812             \exp_not:V \l__zrefclever_refpre_out_tl
1813             \exp_not:N \group_begin:
1814             \exp_not:V \l__zrefclever_reffont_in_tl
1815             % It's two 's', but escaped for DocStrip.
1816             \exp_not:N \hyper@@link
1817             {
1818                 \zref@ifrefcontainsprop {#1} { urluse }
1819                 { \zref@extractdefault {#1} { urluse } {} }

```

```

1820         { \zref@extractdefault {#1} { url } {} }
1821     }
1822     { \zref@extractdefault {#1} { anchor } {} }
1823     {
1824         \exp_not:V \l__zrefclever_refpre_in_tl
1825         \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1826         \exp_not:V \l__zrefclever_refpos_in_tl
1827     }
1828     \exp_not:N \group_end:
1829     \exp_not:V \l__zrefclever_refpos_out_tl
1830     \exp_not:N \group_end:
1831 }
1832 {
1833     \exp_not:N \group_begin:
1834     \exp_not:V \l__zrefclever_reffont_out_tl
1835     \exp_not:V \l__zrefclever_refpre_out_tl
1836     \exp_not:N \group_begin:
1837     \exp_not:V \l__zrefclever_reffont_in_tl
1838     \exp_not:V \l__zrefclever_refpre_in_tl
1839     \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1840     \exp_not:V \l__zrefclever_refpos_in_tl
1841     \exp_not:N \group_end:
1842     \exp_not:V \l__zrefclever_refpos_out_tl
1843     \exp_not:N \group_end:
1844 }
1845 }
1846 { \exp_not:N \zref@default }
1847 }
1848 \cs_generate_variant:Nn \__zrefclever_get_ref:n { V }

```

(End definition for \\_\_zrefclever\_get\_ref:n.)

\\_\_zrefclever\_type\_name\_setup: Auxiliary function to \\_\_zrefclever\_typeset\_refs:. Sets the type name variable \l\_\_zrefclever\_type\_name\_tl. When it cannot be found, clears it.

```

1849 \cs_new_protected:Npn \__zrefclever_type_name_setup:
1850 {
1851     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1852     { \tl_clear:N \l__zrefclever_type_name_tl }
1853     {
1854         \tl_if_empty:nTF \l__zrefclever_type_first_label_type_tl
1855         { \tl_clear:N \l__zrefclever_type_name_tl }
1856         {

```

Determine whether we should use capitalization, abbreviation, and plural.

```

1857     \bool_lazy_or:nnTF
1858     { \l__zrefclever_capitalize_bool }
1859     {
1860         \l__zrefclever_capitalize_first_bool &&
1861         \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1862     }
1863     { \tl_set:Nn \l__zrefclever_name_format_tl {Name} }
1864     { \tl_set:Nn \l__zrefclever_name_format_tl {name} }
1865     % If the queue is empty, we have a singular, otherwise, plural.
1866     \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1867     { \tl_put_right:Nn \l__zrefclever_name_format_tl { -sg } }

```

```

1868     { \tl_put_right:Nn \l__zrefclever_name_format_tl { -pl } }
1869 \bool_lazy_and:nnTF
1870 { \l__zrefclever_abbrev_bool }
1871 {
1872     ! \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 } ||
1873     ! \l__zrefclever_noabbrev_first_bool
1874 }
1875 {
1876     \tl_set:NV \l__zrefclever_name_format_fallback_tl \l__zrefclever_name_format
1877     \tl_put_right:Nn \l__zrefclever_name_format_tl { -ab }
1878 }
1879 { \tl_clear:N \l__zrefclever_name_format_fallback_tl }
1880
1881 \tl_if_empty:NTF \l__zrefclever_name_format_fallback_tl
1882 {
1883     \prop_get:cVNF
1884     { \l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1885       \l__zrefclever_name_format_tl
1886       \l__zrefclever_type_name_tl
1887       {
1888         \__zrefclever_if_transl:xxTF
1889         { \l__zrefclever_ref_language_tl }
1890         {
1891             zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1892             \l__zrefclever_name_format_tl
1893         }
1894         {
1895             \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1896             { \l__zrefclever_ref_language_tl }
1897             {
1898                 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1899                 \l__zrefclever_name_format_tl
1900             }
1901         }
1902         {
1903             \tl_clear:N \l__zrefclever_type_name_tl
1904             \msg_warning:nxx { zref-clever } { missing-name }
1905             { \l__zrefclever_type_first_label_type_tl }
1906         }
1907     }
1908 }
1909 {
1910     \prop_get:cVNF
1911     { \l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1912       \l__zrefclever_name_format_tl
1913       \l__zrefclever_type_name_tl
1914       {
1915         \prop_get:cVNF
1916         { \l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options
1917           \l__zrefclever_name_format_fallback_tl
1918           \l__zrefclever_type_name_tl
1919           {
1920             \__zrefclever_if_transl:xxTF
1921             { \l__zrefclever_ref_language_tl }

```

```

1922         {
1923             zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1924             \l__zrefclever_name_format_tl
1925         }
1926     {
1927         \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1928         { \l__zrefclever_ref_language_tl }
1929         {
1930             zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1931             \l__zrefclever_name_format_tl
1932         }
1933     }
1934     {
1935         \__zrefclever_if_transl:xxTF
1936         { \l__zrefclever_ref_language_tl }
1937         {
1938             zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1939             \l__zrefclever_name_format_fallback_tl
1940         }
1941         {
1942             \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1943             { \l__zrefclever_ref_language_tl }
1944             {
1945                 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1946                 \l__zrefclever_name_format_fallback_tl
1947             }
1948         }
1949         {
1950             \tl_clear:N \l__zrefclever_type_name_tl
1951             \msg_warning:nxx { zref-clever } { missing-name }
1952             { \l__zrefclever_type_first_label_type_tl }
1953         }
1954     }
1955 }
1956 }
1957 }
1958 }
1959 }

```

Signal whether the type name is to be included in the hyperlink or not.

```

1960 \bool_lazy_any:nTF
1961 {
1962     { ! \l__zrefclever_use_hyperref_bool }
1963     { \l__zrefclever_link_star_bool }
1964     { \tl_if_empty_p:N \l__zrefclever_type_name_tl }
1965     { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { false } }
1966 }
1967 { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1968 {
1969     \bool_lazy_any:nTF
1970     {
1971         { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { true } }
1972         {
1973             \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { tsingle } &&
1974             \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl

```



```

1975     }
1976     {
1977         \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { single } &&
1978         \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl &&
1979         \l__zrefclever_typeset_last_bool &&
1980         \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1981     }
1982 }
1983 { \bool_set_true:N \l__zrefclever_name_in_link_bool }
1984 { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1985 }
1986 }

```

(End definition for \\_\_zrefclever\_type\_name\_setup:.)

\\_\_zrefclever\_get\_ref\_first: Auxiliary function to \\_\_zrefclever\_typeset\_refs:. Handles a complete “ref-block”, including “pre” and “pos” elements, *hyperlinking*, and the reference type “name”. For use on the first reference of each type.

```

1987 \cs_new:Npn \__zrefclever_get_ref_first:
1988 {
1989     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1990     { \exp_not:N \zref@default }
1991     {
1992         \bool_if:NTF \l__zrefclever_name_in_link_bool
1993         {
1994             \zref@ifrefcontainsprop
1995             { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
1996             {
1997                 % It's two 's', but escaped for DocStrip.
1998                 \exp_not:N \hyper@@link
1999                 {
2000                     \zref@ifrefcontainsprop
2001                     { \l__zrefclever_type_first_label_tl } { urluse }
2002                     {
2003                         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2004                         { urluse } {}
2005                     }
2006                     {
2007                         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2008                         { url } {}
2009                     }
2010                 }
2011             }
2012             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2013             { anchor } {}
2014         }
2015         {
2016             \exp_not:N \group_begin:
2017             \exp_not:V \l__zrefclever_namefont_tl
2018             \exp_not:V \l__zrefclever_type_name_tl
2019             \exp_not:N \group_end:
2020             \exp_not:V \l__zrefclever_namesep_tl
2021             \exp_not:N \group_begin:
2022             \exp_not:V \l__zrefclever_reffont_out_tl

```

```

2023         \exp_not:V \l__zrefclever_refpre_out_tl
2024         \exp_not:N \group_begin:
2025         \exp_not:V \l__zrefclever_reffont_in_tl
2026         \exp_not:V \l__zrefclever_refpre_in_tl
2027         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2028         { \l__zrefclever_ref_property_tl } {}
2029         \exp_not:V \l__zrefclever_refpos_in_tl
2030         \exp_not:N \group_end:
2031         % hyperlink makes it's own group, we'd like to close the
2032         % 'refpre-out' group after 'refpos-out', but... we close
2033         % it here, and give the trailing 'refpos-out' its own
2034         % group. This will result that formatting given to
2035         % 'refpre-out' will not reach 'refpos-out', but I see no
2036         % alternative, and this has to be handled specially.
2037         \exp_not:N \group_end:
2038     }
2039     \exp_not:N \group_begin:
2040     % Ditto: special treatment.
2041     \exp_not:V \l__zrefclever_reffont_out_tl
2042     \exp_not:V \l__zrefclever_refpos_out_tl
2043     \exp_not:N \group_end:
2044 }
2045 {
2046     \exp_not:N \group_begin:
2047     \exp_not:V \l__zrefclever_namefont_tl
2048     \exp_not:V \l__zrefclever_type_name_tl
2049     \exp_not:N \group_end:
2050     \exp_not:V \l__zrefclever_namesep_tl
2051     \exp_not:N \zref@default
2052 }
2053 }
2054 {
2055     \tl_if_empty:NTF \l__zrefclever_type_name_tl
2056     {
2057         \exp_not:N \zref@default
2058         \exp_not:V \l__zrefclever_namesep_tl
2059     }
2060     {
2061         \exp_not:N \group_begin:
2062         \exp_not:V \l__zrefclever_namefont_tl
2063         \exp_not:V \l__zrefclever_type_name_tl
2064         \exp_not:N \group_end:
2065         \exp_not:V \l__zrefclever_namesep_tl
2066     }
2067     \zref@ifrefcontainsprop
2068     { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
2069     {
2070         \bool_if:nTF
2071         {
2072             \l__zrefclever_use_hyperref_bool &&
2073             ! \l__zrefclever_link_star_bool
2074         }
2075         {
2076             \exp_not:N \group_begin:

```

```

2077 \exp_not:V \l__zrefclever_reffont_out_tl
2078 \exp_not:V \l__zrefclever_refpre_out_tl
2079 \exp_not:N \group_begin:
2080 \exp_not:V \l__zrefclever_reffont_in_tl
2081 % It's two '@s', but escaped for DocStrip.
2082 \exp_not:N \hyper@@link
2083 {
2084   \zref@ifrefcontainsprop
2085   { \l__zrefclever_type_first_label_tl } { urluse }
2086   {
2087     \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2088     { urluse } {}
2089   }
2090   {
2091     \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2092     { url } {}
2093   }
2094 }
2095 {
2096   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2097   { anchor } {}
2098 }
2099 {
2100   \exp_not:V \l__zrefclever_refpre_in_tl
2101   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2102   { \l__zrefclever_ref_property_tl } {}
2103   \exp_not:V \l__zrefclever_refpos_in_tl
2104 }
2105 \exp_not:N \group_end:
2106 \exp_not:V \l__zrefclever_refpos_out_tl
2107 \exp_not:N \group_end:
2108 }
2109 {
2110   \exp_not:N \group_begin:
2111   \exp_not:V \l__zrefclever_reffont_out_tl
2112   \exp_not:V \l__zrefclever_refpre_out_tl
2113   \exp_not:N \group_begin:
2114   \exp_not:V \l__zrefclever_reffont_in_tl
2115   \exp_not:V \l__zrefclever_refpre_in_tl
2116   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2117   { \l__zrefclever_ref_property_tl } {}
2118   \exp_not:V \l__zrefclever_refpos_in_tl
2119   \exp_not:N \group_end:
2120   \exp_not:V \l__zrefclever_refpos_out_tl
2121   \exp_not:N \group_end:
2122 }
2123 }
2124 { \exp_not:N \zref@default }
2125 }
2126 }
2127 }

```

(End definition for `\__zrefclever_get_ref_first:`)

\\_zrefclever\_get\_option\_with\_transl:nN

```

2128 % \Arg{option} \Arg{var to store result}
2129 \cs_new_protected:Npn \__zrefclever_get_option_with_transl:nN #1#2
2130 {
2131   % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.
2132   \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2133   {
2134     % If not found, try the type specific options.
2135     \bool_lazy_all:nTF
2136     {
2137       { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2138       {
2139         \prop_if_exist_p:c
2140         { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2141       }
2142       {
2143         \prop_if_in_p:cn
2144         { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1}
2145       }
2146     }
2147     {
2148       \prop_get:cnN
2149       { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2150     }
2151     {
2152       % If not found, try the type specific translations.
2153       \__zrefclever_if_transl:xxTF
2154       { \l__zrefclever_ref_language_tl }
2155       { zrefclever-type- \l__zrefclever_label_type_a_tl - #1 }
2156       {
2157         \__zrefclever_get_transl:nxx {#2}
2158         { \l__zrefclever_ref_language_tl }
2159         { zrefclever-type- \l__zrefclever_label_type_a_tl - #1 }
2160       }
2161       {
2162         % If not found, try general translations. We are not
2163         % controlling for their existence, but we must make sure all
2164         % options being retrieved with
2165         % \cs{l__zrefclever_get_option_with_transl:nN} have their values set for
2166         % 'English' and 'fallback'.
2167         \__zrefclever_get_transl:nxx {#2}
2168         { \l__zrefclever_ref_language_tl }
2169         { zrefclever-default- #1 }
2170       }
2171     }
2172   }
2173 }

```

(End definition for \\_\_zrefclever\_get\_option\_with\_transl:nN.)

\\_\_zrefclever\_get\_option\_plain:nN

```

2174 \cs_new_protected:Npn \__zrefclever_get_option_plain:nN #1#2
2175 {
2176   % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.

```

```

2177 \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2178 {
2179   % If not found, try the type specific options.
2180   \bool_lazy_and:nnTF
2181     { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2182     {
2183       \prop_if_exist_p:c
2184         { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2185     }
2186     {
2187       \prop_get:cnNF
2188         { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2189         { \tl_clear:N #2 }
2190     }
2191     { \tl_clear:N #2 }
2192   }
2193 }

```

(End definition for \\_\_zrefclever\_get\_option\_plain:nN.)

\\_\_zrefclever\_labels\_in\_sequence:nn Sets \l\_\_zrefclever\_next\_maybe\_range\_bool to true if label ‘1’ comes in immediate sequence from label ‘2’. And sets both \l\_\_zrefclever\_next\_maybe\_range\_bool and \l\_\_zrefclever\_next\_is\_same\_bool if the labels are the “same”.

```

2194 \cs_new_protected:Npn \__zrefclever_labels_in_sequence:nn #1#2
2195 {
2196   \bool_if:NTF \l__zrefclever_page_ref_bool
2197   {
2198     \exp_args:Nxx \tl_if_eq:nnT
2199       { \zref@extractdefault {#1} { zc@pgfmt } { } }
2200       { \zref@extractdefault {#2} { zc@pgfmt } { } }
2201     {
2202       \int_compare:nNnTF
2203         { \zref@extractdefault {#1} { zc@pgval } {-2} + 1 }
2204         =
2205         { \zref@extractdefault {#2} { zc@pgval } {-1} }
2206         { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2207         {
2208           \int_compare:nNnT
2209             { \zref@extractdefault {#1} { zc@pgval } {-1} }
2210             =
2211             { \zref@extractdefault {#2} { zc@pgval } {-1} }
2212           {
2213             \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2214             \bool_set_true:N \l__zrefclever_next_is_same_bool
2215           }
2216         }
2217     }
2218   }
2219   {
2220     \exp_args:Nxx \tl_if_eq:nnT
2221       { \zref@extractdefault {#1} { counter } { } }
2222       { \zref@extractdefault {#2} { counter } { } }
2223     {
2224       \exp_args:Nxx \tl_if_eq:nnT

```

```

2225 { \zref@extractdefault {#1} { zc@enclval } { } }
2226 { \zref@extractdefault {#2} { zc@enclval } { } }
2227 {
2228   \int_compare:nNnTF
2229     { \zref@extractdefault {#1} { zc@cntval } {-2} + 1 }
2230     =
2231     { \zref@extractdefault {#2} { zc@cntval } {-1} }
2232     { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2233     {
2234       \int_compare:nNnT
2235         { \zref@extractdefault {#1} { zc@cntval } {-1} }
2236         =
2237         { \zref@extractdefault {#2} { zc@cntval } {-1} }
2238         {
2239           \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2240           \bool_set_true:N \l__zrefclever_next_is_same_bool
2241         }
2242       }
2243     }
2244   }
2245 }
2246 }

```

(End definition for `\__zrefclever_labels_in_sequence:nn`.)

## 10 Special handling

This section is meant to aggregate any “special handling” needed for L<sup>A</sup>T<sub>E</sub>X kernel features, document classes, and packages, needed for `zref-clever` to work properly with them. It is not meant to be a “kitchen sink of workarounds”. Rather, I intend to keep this as lean as possible, trying to add things selectively when they are safe and reasonable. And, hopefully, doing so by proper setting of `zref-clever`’s options, not by messing with other packages’ code. In particular, I do not mean to compensate for “lack of support for `zref`” by individual packages here, unless there is really no alternative.

### 10.1 Appendix

Another relevant use case of the same general problem of different types for the same counter is the `\appendix` which in some document classes, including the standard ones, change the sectioning commands looks but, of course, keep using the same counter (`book.cls` and `report.cls` reset counters `chapter` and `section` to 0, change `\@chapapp` to use `\appendixname` and use `\@Alph` for `\thechapter`; `article.cls` resets counters `section` and `subsection` to 0, and uses `\@Alph` for `\thesection`; `memoir.cls`, `scrbook.cls` and `scrarticle.cls` do the same as their corresponding standard classes, and sometimes a little more, but what interests us here is pretty much the same; see also the `appendix` package).

## 10.2 `\newtheorem`

## 10.3 `enumitem` package

TODO Option `counterresetby` should probably be extended for `enumitem`, conditioned on it being loaded.

# 11 Translations

## Fallback

All options retrieved with `\__zrefclever_get_option_with_transl:nN` must have their values set for ‘fallback’, since this is what will be retrieved if `babel` or `polyglossia` is loaded and sets a language which `zref-clever` does not know. On the other hand type-specific options are not looked for in ‘fallback’.

```
2247 \__zrefclever_declare_default_transl:nnn { fallback } { namesep } {\nobreakspace}
2248 \__zrefclever_declare_default_transl:nnn { fallback } { pairsep } { {,~}
2249 \__zrefclever_declare_default_transl:nnn { fallback } { listsep } { {,~}
2250 \__zrefclever_declare_default_transl:nnn { fallback } { lastsep } { {,~}
2251 \__zrefclever_declare_default_transl:nnn { fallback } { tpairsep } { {,~}
2252 \__zrefclever_declare_default_transl:nnn { fallback } { tlistsep } { {,~}
2253 \__zrefclever_declare_default_transl:nnn { fallback } { tlastsep } { {,~}
2254 \__zrefclever_declare_default_transl:nnn { fallback } { notesep } { {~}
2255 \__zrefclever_declare_default_transl:nnn { fallback } { rangeseq } {\textendash}
2256 \__zrefclever_declare_default_transl:nnn { fallback } { refpre } {}
2257 \__zrefclever_declare_default_transl:nnn { fallback } { refpos } {}
2258 \__zrefclever_declare_default_transl:nnn { fallback } { refpre-in } {}
2259 \__zrefclever_declare_default_transl:nnn { fallback } { refpos-in } {}

2260 \</package>
2261 \<*lang-english>
```

## English

All options retrieved with `\__zrefclever_get_option_with_transl:nN` must have their values set for ‘English’, since this is what will be retrieved if no language package is loaded.

```
2262 \ProvideDictionaryFor{English}{zref-clever}
2263
2264 \zcDicDefaultTransl{namesep}{\nobreakspace}
2265 \zcDicDefaultTransl{pairsep}{~and\nobreakspace}
2266 \zcDicDefaultTransl{listsep}{~,~}
2267 \zcDicDefaultTransl{lastsep}{~and\nobreakspace}
2268 \zcDicDefaultTransl{tpairsep}{~and\nobreakspace}
2269 \zcDicDefaultTransl{tlistsep}{~,~}
2270 \zcDicDefaultTransl{tlastsep}{~,~and\nobreakspace}
2271 \zcDicDefaultTransl{notesep}{~}
2272 \zcDicDefaultTransl{rangeseq}{~to\nobreakspace}
2273 \zcDicDefaultTransl{refpre}{}
2274 \zcDicDefaultTransl{refpos}{}
2275 \zcDicDefaultTransl{refpre-in}{}
2276 \zcDicDefaultTransl{refpos-in}{}
2277
2278 \zcDicTypeTransl{part}{Name-sg}{Part}
```

2279 \zcDicTypeTransl{part}{name-sg}{part}  
2280 \zcDicTypeTransl{part}{Name-pl}{Parts}  
2281 \zcDicTypeTransl{part}{name-pl}{parts}  
2282  
2283 \zcDicTypeTransl{chapter}{Name-sg}{Chapter}  
2284 \zcDicTypeTransl{chapter}{name-sg}{chapter}  
2285 \zcDicTypeTransl{chapter}{Name-pl}{Chapters}  
2286 \zcDicTypeTransl{chapter}{name-pl}{chapters}  
2287  
2288 \zcDicTypeTransl{section}{Name-sg}{Section}  
2289 \zcDicTypeTransl{section}{name-sg}{section}  
2290 \zcDicTypeTransl{section}{Name-pl}{Sections}  
2291 \zcDicTypeTransl{section}{name-pl}{sections}  
2292  
2293 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraph}  
2294 \zcDicTypeTransl{paragraph}{name-sg}{paragraph}  
2295 \zcDicTypeTransl{paragraph}{Name-pl}{Paragraphs}  
2296 \zcDicTypeTransl{paragraph}{name-pl}{paragraphs}  
2297 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}  
2298 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}  
2299 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}  
2300 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}  
2301  
2302 \zcDicTypeTransl{appendix}{Name-sg}{Appendix}  
2303 \zcDicTypeTransl{appendix}{name-sg}{appendix}  
2304 \zcDicTypeTransl{appendix}{Name-pl}{Appendices}  
2305 \zcDicTypeTransl{appendix}{name-pl}{appendices}  
2306  
2307 \zcDicTypeTransl{page}{Name-sg}{Page}  
2308 \zcDicTypeTransl{page}{name-sg}{page}  
2309 \zcDicTypeTransl{page}{Name-pl}{Pages}  
2310 \zcDicTypeTransl{page}{name-pl}{pages}  
2311 \zcDicTypeTransl{page}{name-sg-ab}{p.}  
2312 \zcDicTypeTransl{page}{name-pl-ab}{pp.}  
2313  
2314 \zcDicTypeTransl{line}{Name-sg}{Line}  
2315 \zcDicTypeTransl{line}{name-sg}{line}  
2316 \zcDicTypeTransl{line}{Name-pl}{Lines}  
2317 \zcDicTypeTransl{line}{name-pl}{lines}  
2318  
2319 \zcDicTypeTransl{figure}{Name-sg}{Figure}  
2320 \zcDicTypeTransl{figure}{name-sg}{figure}  
2321 \zcDicTypeTransl{figure}{Name-pl}{Figures}  
2322 \zcDicTypeTransl{figure}{name-pl}{figures}  
2323 \zcDicTypeTransl{figure}{Name-sg-ab}{Fig.}  
2324 \zcDicTypeTransl{figure}{name-sg-ab}{fig.}  
2325 \zcDicTypeTransl{figure}{Name-pl-ab}{Figs.}  
2326 \zcDicTypeTransl{figure}{name-pl-ab}{figs.}  
2327  
2328 \zcDicTypeTransl{table}{Name-sg}{Table}  
2329 \zcDicTypeTransl{table}{name-sg}{table}  
2330 \zcDicTypeTransl{table}{Name-pl}{Tables}  
2331 \zcDicTypeTransl{table}{name-pl}{tables}  
2332



2333 \zcDicTypeTransl{item}{Name-sg}{Item}  
2334 \zcDicTypeTransl{item}{name-sg}{item}  
2335 \zcDicTypeTransl{item}{Name-pl}{Items}  
2336 \zcDicTypeTransl{item}{name-pl}{items}  
2337  
2338 \zcDicTypeTransl{footnote}{Name-sg}{Footnote}  
2339 \zcDicTypeTransl{footnote}{name-sg}{footnote}  
2340 \zcDicTypeTransl{footnote}{Name-pl}{Footnotes}  
2341 \zcDicTypeTransl{footnote}{name-pl}{footnotes}  
2342  
2343 \zcDicTypeTransl{note}{Name-sg}{Note}  
2344 \zcDicTypeTransl{note}{name-sg}{note}  
2345 \zcDicTypeTransl{note}{Name-pl}{Notes}  
2346 \zcDicTypeTransl{note}{name-pl}{notes}  
2347  
2348 \zcDicTypeTransl{equation}{Name-sg}{Equation}  
2349 \zcDicTypeTransl{equation}{name-sg}{equation}  
2350 \zcDicTypeTransl{equation}{Name-pl}{Equations}  
2351 \zcDicTypeTransl{equation}{name-pl}{equations}  
2352 \zcDicTypeTransl{equation}{Name-sg-ab}{Eq.}  
2353 \zcDicTypeTransl{equation}{name-sg-ab}{eq.}  
2354 \zcDicTypeTransl{equation}{Name-pl-ab}{Eqs.}  
2355 \zcDicTypeTransl{equation}{name-pl-ab}{eqs.}  
2356 \zcDicTypeTransl{equation}{refpre-in}{(}  
2357 \zcDicTypeTransl{equation}{refpos-in}{)}  
2358  
2359 \zcDicTypeTransl{theorem}{Name-sg}{Theorem}  
2360 \zcDicTypeTransl{theorem}{name-sg}{theorem}  
2361 \zcDicTypeTransl{theorem}{Name-pl}{Theorems}  
2362 \zcDicTypeTransl{theorem}{name-pl}{theorems}  
2363  
2364 \zcDicTypeTransl{lemma}{Name-sg}{Lemma}  
2365 \zcDicTypeTransl{lemma}{name-sg}{lemma}  
2366 \zcDicTypeTransl{lemma}{Name-pl}{Lemmas}  
2367 \zcDicTypeTransl{lemma}{name-pl}{lemmas}  
2368  
2369 \zcDicTypeTransl{corollary}{Name-sg}{Corollary}  
2370 \zcDicTypeTransl{corollary}{name-sg}{corollary}  
2371 \zcDicTypeTransl{corollary}{Name-pl}{Corollaries}  
2372 \zcDicTypeTransl{corollary}{name-pl}{corollaries}  
2373  
2374 \zcDicTypeTransl{proposition}{Name-sg}{Proposition}  
2375 \zcDicTypeTransl{proposition}{name-sg}{proposition}  
2376 \zcDicTypeTransl{proposition}{Name-pl}{Propositions}  
2377 \zcDicTypeTransl{proposition}{name-pl}{propositions}  
2378  
2379 \zcDicTypeTransl{definition}{Name-sg}{Definition}  
2380 \zcDicTypeTransl{definition}{name-sg}{definition}  
2381 \zcDicTypeTransl{definition}{Name-pl}{Definitions}  
2382 \zcDicTypeTransl{definition}{name-pl}{definitions}  
2383  
2384 \zcDicTypeTransl{proof}{Name-sg}{Proof}  
2385 \zcDicTypeTransl{proof}{name-sg}{proof}  
2386 \zcDicTypeTransl{proof}{Name-pl}{Proofs}

2387 \zcDicTypeTransl{proof}{name-pl}{proofs}  
 2388  
 2389 \zcDicTypeTransl{result}{Name-sg}{Result}  
 2390 \zcDicTypeTransl{result}{name-sg}{result}  
 2391 \zcDicTypeTransl{result}{Name-pl}{Results}  
 2392 \zcDicTypeTransl{result}{name-pl}{results}  
 2393  
 2394 \zcDicTypeTransl{example}{Name-sg}{Example}  
 2395 \zcDicTypeTransl{example}{name-sg}{example}  
 2396 \zcDicTypeTransl{example}{Name-pl}{Examples}  
 2397 \zcDicTypeTransl{example}{name-pl}{examples}  
 2398  
 2399 \zcDicTypeTransl{remark}{Name-sg}{Remark}  
 2400 \zcDicTypeTransl{remark}{name-sg}{remark}  
 2401 \zcDicTypeTransl{remark}{Name-pl}{Remarks}  
 2402 \zcDicTypeTransl{remark}{name-pl}{remarks}  
 2403  
 2404 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithm}  
 2405 \zcDicTypeTransl{algorithm}{name-sg}{algorithm}  
 2406 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithms}  
 2407 \zcDicTypeTransl{algorithm}{name-pl}{algorithms}  
 2408  
 2409 \zcDicTypeTransl{listing}{Name-sg}{Listing}  
 2410 \zcDicTypeTransl{listing}{name-sg}{listing}  
 2411 \zcDicTypeTransl{listing}{Name-pl}{Listings}  
 2412 \zcDicTypeTransl{listing}{name-pl}{listings}  
 2413  
 2414 \zcDicTypeTransl{exercise}{Name-sg}{Exercise}  
 2415 \zcDicTypeTransl{exercise}{name-sg}{exercise}  
 2416 \zcDicTypeTransl{exercise}{Name-pl}{Exercises}  
 2417 \zcDicTypeTransl{exercise}{name-pl}{exercises}  
 2418  
 2419 \zcDicTypeTransl{solution}{Name-sg}{Solution}  
 2420 \zcDicTypeTransl{solution}{name-sg}{solution}  
 2421 \zcDicTypeTransl{solution}{Name-pl}{Solutions}  
 2422 \zcDicTypeTransl{solution}{name-pl}{solutions}  
 2423  $\langle$ /lang-english)  
 2424  $\langle$ \*lang-german)

## German

2425 \ProvideDictionaryFor{German}{zref-clever}  
 2426  
 2427 \zcDicDefaultTransl{namesep}{\nobreakspace}  
 2428 \zcDicDefaultTransl{pairsep}{\sim\und\nobreakspace}  
 2429 \zcDicDefaultTransl{listsep}{\{, \sim}  
 2430 \zcDicDefaultTransl{lastsep}{\sim\und\nobreakspace}  
 2431 \zcDicDefaultTransl{tpairsep}{\sim\und\nobreakspace}  
 2432 \zcDicDefaultTransl{tlistsep}{\{, \sim}  
 2433 \zcDicDefaultTransl{tlastsep}{\sim\und\nobreakspace}  
 2434 \zcDicDefaultTransl{notesep}{\{~}  
 2435 \zcDicDefaultTransl{rangesep}{\sim\und\nobreakspace}  
 2436

2437 \zcDicTypeTransl{part}{Name-sg}{Teil}  
 2438 \zcDicTypeTransl{part}{name-sg}{Teil}  
 2439 \zcDicTypeTransl{part}{Name-pl}{Teile}  
 2440 \zcDicTypeTransl{part}{name-pl}{Teile}  
 2441  
 2442 \zcDicTypeTransl{chapter}{Name-sg}{Kapitel}  
 2443 \zcDicTypeTransl{chapter}{name-sg}{Kapitel}  
 2444 \zcDicTypeTransl{chapter}{Name-pl}{Kapitel}  
 2445 \zcDicTypeTransl{chapter}{name-pl}{Kapitel}  
 2446  
 2447 \zcDicTypeTransl{section}{Name-sg}{Abschnitt}  
 2448 \zcDicTypeTransl{section}{name-sg}{Abschnitt}  
 2449 \zcDicTypeTransl{section}{Name-pl}{Abschnitte}  
 2450 \zcDicTypeTransl{section}{name-pl}{Abschnitte}  
 2451  
 2452 \zcDicTypeTransl{paragraph}{Name-sg}{Absatz}  
 2453 \zcDicTypeTransl{paragraph}{name-sg}{Absatz}  
 2454 \zcDicTypeTransl{paragraph}{Name-pl}{Absätze}  
 2455 \zcDicTypeTransl{paragraph}{name-pl}{Absätze}  
 2456  
 2457 \zcDicTypeTransl{appendix}{Name-sg}{Anhang}  
 2458 \zcDicTypeTransl{appendix}{name-sg}{Anhang}  
 2459 \zcDicTypeTransl{appendix}{Name-pl}{Anhänge}  
 2460 \zcDicTypeTransl{appendix}{name-pl}{Anhänge}  
 2461  
 2462 \zcDicTypeTransl{page}{Name-sg}{Seite}  
 2463 \zcDicTypeTransl{page}{name-sg}{Seite}  
 2464 \zcDicTypeTransl{page}{Name-pl}{Seiten}  
 2465 \zcDicTypeTransl{page}{name-pl}{Seiten}  
 2466  
 2467 \zcDicTypeTransl{line}{Name-sg}{Zeile}  
 2468 \zcDicTypeTransl{line}{name-sg}{Zeile}  
 2469 \zcDicTypeTransl{line}{Name-pl}{Zeilen}  
 2470 \zcDicTypeTransl{line}{name-pl}{Zeilen}  
 2471  
 2472 \zcDicTypeTransl{figure}{Name-sg}{Abbildung}  
 2473 \zcDicTypeTransl{figure}{name-sg}{Abbildung}  
 2474 \zcDicTypeTransl{figure}{Name-pl}{Abbildungen}  
 2475 \zcDicTypeTransl{figure}{name-pl}{Abbildungen}  
 2476 \zcDicTypeTransl{figure}{Name-sg-ab}{Abb.}  
 2477 \zcDicTypeTransl{figure}{name-sg-ab}{Abb.}  
 2478 \zcDicTypeTransl{figure}{Name-pl-ab}{Abb.}  
 2479 \zcDicTypeTransl{figure}{name-pl-ab}{Abb.}  
 2480  
 2481 \zcDicTypeTransl{table}{Name-sg}{Tabelle}  
 2482 \zcDicTypeTransl{table}{name-sg}{Tabelle}  
 2483 \zcDicTypeTransl{table}{Name-pl}{Tabellen}  
 2484 \zcDicTypeTransl{table}{name-pl}{Tabellen}  
 2485  
 2486 \zcDicTypeTransl{item}{Name-sg}{Punkt}  
 2487 \zcDicTypeTransl{item}{name-sg}{Punkt}  
 2488 \zcDicTypeTransl{item}{Name-pl}{Punkte}  
 2489 \zcDicTypeTransl{item}{name-pl}{Punkte}  
 2490

2491 \zcDicTypeTransl{footnote}{Name-sg}{Fußnote}  
 2492 \zcDicTypeTransl{footnote}{name-sg}{Fußnote}  
 2493 \zcDicTypeTransl{footnote}{Name-pl}{Fußnoten}  
 2494 \zcDicTypeTransl{footnote}{name-pl}{Fußnoten}  
 2495  
 2496 \zcDicTypeTransl{note}{Name-sg}{Anmerkung}  
 2497 \zcDicTypeTransl{note}{name-sg}{Anmerkung}  
 2498 \zcDicTypeTransl{note}{Name-pl}{Anmerkungen}  
 2499 \zcDicTypeTransl{note}{name-pl}{Anmerkungen}  
 2500  
 2501 \zcDicTypeTransl{equation}{Name-sg}{Gleichung}  
 2502 \zcDicTypeTransl{equation}{name-sg}{Gleichung}  
 2503 \zcDicTypeTransl{equation}{Name-pl}{Gleichungen}  
 2504 \zcDicTypeTransl{equation}{name-pl}{Gleichungen}  
 2505 \zcDicTypeTransl{equation}{refpre-in}{(}  
 2506 \zcDicTypeTransl{equation}{refpos-in}{)}  
 2507  
 2508 \zcDicTypeTransl{theorem}{Name-sg}{Theorem}  
 2509 \zcDicTypeTransl{theorem}{name-sg}{Theorem}  
 2510 \zcDicTypeTransl{theorem}{Name-pl}{Theoreme}  
 2511 \zcDicTypeTransl{theorem}{name-pl}{Theoreme}  
 2512  
 2513 \zcDicTypeTransl{lemma}{Name-sg}{Lemma}  
 2514 \zcDicTypeTransl{lemma}{name-sg}{Lemma}  
 2515 \zcDicTypeTransl{lemma}{Name-pl}{Lemmata}  
 2516 \zcDicTypeTransl{lemma}{name-pl}{Lemmata}  
 2517  
 2518 \zcDicTypeTransl{corollary}{Name-sg}{Korollar}  
 2519 \zcDicTypeTransl{corollary}{name-sg}{Korollar}  
 2520 \zcDicTypeTransl{corollary}{Name-pl}{Korollare}  
 2521 \zcDicTypeTransl{corollary}{name-pl}{Korollare}  
 2522  
 2523 \zcDicTypeTransl{proposition}{Name-sg}{Satz}  
 2524 \zcDicTypeTransl{proposition}{name-sg}{Satz}  
 2525 \zcDicTypeTransl{proposition}{Name-pl}{Sätze}  
 2526 \zcDicTypeTransl{proposition}{name-pl}{Sätze}  
 2527  
 2528 \zcDicTypeTransl{definition}{Name-sg}{Definition}  
 2529 \zcDicTypeTransl{definition}{name-sg}{Definition}  
 2530 \zcDicTypeTransl{definition}{Name-pl}{Definitionen}  
 2531 \zcDicTypeTransl{definition}{name-pl}{Definitionen}  
 2532  
 2533 \zcDicTypeTransl{proof}{Name-sg}{Beweis}  
 2534 \zcDicTypeTransl{proof}{name-sg}{Beweis}  
 2535 \zcDicTypeTransl{proof}{Name-pl}{Beweise}  
 2536 \zcDicTypeTransl{proof}{name-pl}{Beweise}  
 2537  
 2538 \zcDicTypeTransl{result}{Name-sg}{Ergebnis}  
 2539 \zcDicTypeTransl{result}{name-sg}{Ergebnis}  
 2540 \zcDicTypeTransl{result}{Name-pl}{Ergebnisse}  
 2541 \zcDicTypeTransl{result}{name-pl}{Ergebnisse}  
 2542  
 2543 \zcDicTypeTransl{example}{Name-sg}{Beispiel}  
 2544 \zcDicTypeTransl{example}{name-sg}{Beispiel}

```

2545 \zcDicTypeTransl{example}{Name-pl}{Beispiele}
2546 \zcDicTypeTransl{example}{name-pl}{Beispiele}
2547
2548 \zcDicTypeTransl{remark}{Name-sg}{Bemerkung}
2549 \zcDicTypeTransl{remark}{name-sg}{Bemerkung}
2550 \zcDicTypeTransl{remark}{Name-pl}{Bemerkungen}
2551 \zcDicTypeTransl{remark}{name-pl}{Bemerkungen}
2552
2553 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithmus}
2554 \zcDicTypeTransl{algorithm}{name-sg}{Algorithmus}
2555 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithmen}
2556 \zcDicTypeTransl{algorithm}{name-pl}{Algorithmen}
2557
2558 \zcDicTypeTransl{listing}{Name-sg}{Listing} % CHECK
2559 \zcDicTypeTransl{listing}{name-sg}{Listing} % CHECK
2560 \zcDicTypeTransl{listing}{Name-pl}{Listings} % CHECK
2561 \zcDicTypeTransl{listing}{name-pl}{Listings} % CHECK
2562
2563 \zcDicTypeTransl{exercise}{Name-sg}{Übungsaufgabe}
2564 \zcDicTypeTransl{exercise}{name-sg}{Übungsaufgabe}
2565 \zcDicTypeTransl{exercise}{Name-pl}{Übungsaufgaben}
2566 \zcDicTypeTransl{exercise}{name-pl}{Übungsaufgaben}
2567
2568 \zcDicTypeTransl{solution}{Name-sg}{Lösung}
2569 \zcDicTypeTransl{solution}{name-sg}{Lösung}
2570 \zcDicTypeTransl{solution}{Name-pl}{Lösungen}
2571 \zcDicTypeTransl{solution}{name-pl}{Lösungen}
2572 </lang-german>
2573 <*lang-french>

```

## French

```

2574 \ProvideDictionaryFor{French}{zref-clever}
2575
2576 \zcDicDefaultTransl{namesep}{\nobreakspace}
2577 \zcDicDefaultTransl{pairsep}{~et\nobreakspace}
2578 \zcDicDefaultTransl{listsep}{,~}
2579 \zcDicDefaultTransl{lastsep}{~et\nobreakspace}
2580 \zcDicDefaultTransl{tpairsep}{~et\nobreakspace}
2581 \zcDicDefaultTransl{tlistsep}{,~}
2582 \zcDicDefaultTransl{tlastsep}{~et\nobreakspace}
2583 \zcDicDefaultTransl{notesep}{~}
2584 \zcDicDefaultTransl{rangesep}{~à\nobreakspace}
2585
2586 \zcDicTypeTransl{part}{Name-sg}{Partie}
2587 \zcDicTypeTransl{part}{name-sg}{partie}
2588 \zcDicTypeTransl{part}{Name-pl}{Parties}
2589 \zcDicTypeTransl{part}{name-pl}{parties}
2590
2591 \zcDicTypeTransl{chapter}{Name-sg}{Chapitre}
2592 \zcDicTypeTransl{chapter}{name-sg}{chapitre}
2593 \zcDicTypeTransl{chapter}{Name-pl}{Chapitres}
2594 \zcDicTypeTransl{chapter}{name-pl}{chapitres}
2595

```

2596 \zcDicTypeTransl{section}{Name-sg}{Section}  
 2597 \zcDicTypeTransl{section}{name-sg}{section}  
 2598 \zcDicTypeTransl{section}{Name-pl}{Sections}  
 2599 \zcDicTypeTransl{section}{name-pl}{sections}  
 2600  
 2601 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraphe}  
 2602 \zcDicTypeTransl{paragraph}{name-sg}{paragraphe}  
 2603 \zcDicTypeTransl{paragraph}{Name-pl}{Paragraphes}  
 2604 \zcDicTypeTransl{paragraph}{name-pl}{paragraphes}  
 2605  
 2606 \zcDicTypeTransl{appendix}{Name-sg}{Annexe}  
 2607 \zcDicTypeTransl{appendix}{name-sg}{annexe}  
 2608 \zcDicTypeTransl{appendix}{Name-pl}{Annexes}  
 2609 \zcDicTypeTransl{appendix}{name-pl}{annexes}  
 2610  
 2611 \zcDicTypeTransl{page}{Name-sg}{Page}  
 2612 \zcDicTypeTransl{page}{name-sg}{page}  
 2613 \zcDicTypeTransl{page}{Name-pl}{Pages}  
 2614 \zcDicTypeTransl{page}{name-pl}{pages}  
 2615  
 2616 \zcDicTypeTransl{line}{Name-sg}{Ligne}  
 2617 \zcDicTypeTransl{line}{name-sg}{ligne}  
 2618 \zcDicTypeTransl{line}{Name-pl}{Lignes}  
 2619 \zcDicTypeTransl{line}{name-pl}{lignes}  
 2620  
 2621 \zcDicTypeTransl{figure}{Name-sg}{Figure}  
 2622 \zcDicTypeTransl{figure}{name-sg}{figure}  
 2623 \zcDicTypeTransl{figure}{Name-pl}{Figures}  
 2624 \zcDicTypeTransl{figure}{name-pl}{figures}  
 2625  
 2626 \zcDicTypeTransl{table}{Name-sg}{Table}  
 2627 \zcDicTypeTransl{table}{name-sg}{table}  
 2628 \zcDicTypeTransl{table}{Name-pl}{Tables}  
 2629 \zcDicTypeTransl{table}{name-pl}{tables}  
 2630  
 2631 \zcDicTypeTransl{item}{Name-sg}{Point}  
 2632 \zcDicTypeTransl{item}{name-sg}{point}  
 2633 \zcDicTypeTransl{item}{Name-pl}{Points}  
 2634 \zcDicTypeTransl{item}{name-pl}{points}  
 2635  
 2636 \zcDicTypeTransl{footnote}{Name-sg}{Note}  
 2637 \zcDicTypeTransl{footnote}{name-sg}{note}  
 2638 \zcDicTypeTransl{footnote}{Name-pl}{Notes}  
 2639 \zcDicTypeTransl{footnote}{name-pl}{notes}  
 2640  
 2641 \zcDicTypeTransl{note}{Name-sg}{Note}  
 2642 \zcDicTypeTransl{note}{name-sg}{note}  
 2643 \zcDicTypeTransl{note}{Name-pl}{Notes}  
 2644 \zcDicTypeTransl{note}{name-pl}{notes}  
 2645  
 2646 \zcDicTypeTransl{equation}{Name-sg}{Équation}  
 2647 \zcDicTypeTransl{equation}{name-sg}{équation}  
 2648 \zcDicTypeTransl{equation}{Name-pl}{Équations}  
 2649 \zcDicTypeTransl{equation}{name-pl}{équations}

2650 \zcDicTypeTransl{equation}{refpre-in}{()  
2651 \zcDicTypeTransl{equation}{refpos-in}{()  
2652  
2653 \zcDicTypeTransl{theorem}{Name-sg}{Théorème}  
2654 \zcDicTypeTransl{theorem}{name-sg}{théorème}  
2655 \zcDicTypeTransl{theorem}{Name-pl}{Théorèmes}  
2656 \zcDicTypeTransl{theorem}{name-pl}{théorèmes}  
2657  
2658 \zcDicTypeTransl{lemma}{Name-sg}{Lemme}  
2659 \zcDicTypeTransl{lemma}{name-sg}{lemme}  
2660 \zcDicTypeTransl{lemma}{Name-pl}{Lemmes}  
2661 \zcDicTypeTransl{lemma}{name-pl}{lemmes}  
2662  
2663 \zcDicTypeTransl{corollary}{Name-sg}{Corollaire}  
2664 \zcDicTypeTransl{corollary}{name-sg}{corollaire}  
2665 \zcDicTypeTransl{corollary}{Name-pl}{Corollaires}  
2666 \zcDicTypeTransl{corollary}{name-pl}{corollaires}  
2667  
2668 \zcDicTypeTransl{proposition}{Name-sg}{Proposition}  
2669 \zcDicTypeTransl{proposition}{name-sg}{proposition}  
2670 \zcDicTypeTransl{proposition}{Name-pl}{Propositions}  
2671 \zcDicTypeTransl{proposition}{name-pl}{propositions}  
2672  
2673 \zcDicTypeTransl{definition}{Name-sg}{Définition}  
2674 \zcDicTypeTransl{definition}{name-sg}{définition}  
2675 \zcDicTypeTransl{definition}{Name-pl}{Définitions}  
2676 \zcDicTypeTransl{definition}{name-pl}{définitions}  
2677  
2678 \zcDicTypeTransl{proof}{Name-sg}{Démonstration}  
2679 \zcDicTypeTransl{proof}{name-sg}{démonstration}  
2680 \zcDicTypeTransl{proof}{Name-pl}{Démonstrations}  
2681 \zcDicTypeTransl{proof}{name-pl}{démonstrations}  
2682  
2683 \zcDicTypeTransl{result}{Name-sg}{Résultat}  
2684 \zcDicTypeTransl{result}{name-sg}{résultat}  
2685 \zcDicTypeTransl{result}{Name-pl}{Résultats}  
2686 \zcDicTypeTransl{result}{name-pl}{résultats}  
2687  
2688 \zcDicTypeTransl{example}{Name-sg}{Exemple}  
2689 \zcDicTypeTransl{example}{name-sg}{exemple}  
2690 \zcDicTypeTransl{example}{Name-pl}{Exemples}  
2691 \zcDicTypeTransl{example}{name-pl}{exemples}  
2692  
2693 \zcDicTypeTransl{remark}{Name-sg}{Remarque}  
2694 \zcDicTypeTransl{remark}{name-sg}{remarque}  
2695 \zcDicTypeTransl{remark}{Name-pl}{Remarques}  
2696 \zcDicTypeTransl{remark}{name-pl}{remarques}  
2697  
2698 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithme}  
2699 \zcDicTypeTransl{algorithm}{name-sg}{algorithme}  
2700 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithmes}  
2701 \zcDicTypeTransl{algorithm}{name-pl}{algorithmes}  
2702  
2703 \zcDicTypeTransl{listing}{Name-sg}{Liste}

```

2704 \zcDicTypeTransl{listing}{name-sg}{liste}
2705 \zcDicTypeTransl{listing}{Name-pl}{Listes}
2706 \zcDicTypeTransl{listing}{name-pl}{listes}
2707
2708 \zcDicTypeTransl{exercise}{Name-sg}{Exercice}
2709 \zcDicTypeTransl{exercise}{name-sg}{exercice}
2710 \zcDicTypeTransl{exercise}{Name-pl}{Exercices}
2711 \zcDicTypeTransl{exercise}{name-pl}{exercices}
2712
2713 \zcDicTypeTransl{solution}{Name-sg}{Solution}
2714 \zcDicTypeTransl{solution}{name-sg}{solution}
2715 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
2716 \zcDicTypeTransl{solution}{name-pl}{solutions}
2717 </lang-french>
2718 <*lang-portuguese>

```

## Portuguese

```

2719 \ProvideDictionaryFor{Portuguese}{zref-clever}
2720
2721 \zcDicDefaultTransl{namesep}{\nobreakspace}
2722 \zcDicDefaultTransl{pairsep}{~e\nobreakspace}
2723 \zcDicDefaultTransl{listsep}{~,~}
2724 \zcDicDefaultTransl{lastsep}{~e\nobreakspace}
2725 \zcDicDefaultTransl{tpairsep}{~e\nobreakspace}
2726 \zcDicDefaultTransl{tlistsep}{~,~}
2727 \zcDicDefaultTransl{tlastsep}{~e\nobreakspace}
2728 \zcDicDefaultTransl{notesep}{~}
2729 \zcDicDefaultTransl{rangesep}{~a\nobreakspace}
2730
2731 \zcDicTypeTransl{part}{Name-sg}{Parte}
2732 \zcDicTypeTransl{part}{name-sg}{parte}
2733 \zcDicTypeTransl{part}{Name-pl}{Partes}
2734 \zcDicTypeTransl{part}{name-pl}{partes}
2735
2736 \zcDicTypeTransl{chapter}{Name-sg}{Capítulo}
2737 \zcDicTypeTransl{chapter}{name-sg}{capítulo}
2738 \zcDicTypeTransl{chapter}{Name-pl}{Capítulos}
2739 \zcDicTypeTransl{chapter}{name-pl}{capítulos}
2740
2741 \zcDicTypeTransl{section}{Name-sg}{Seção}
2742 \zcDicTypeTransl{section}{name-sg}{seção}
2743 \zcDicTypeTransl{section}{Name-pl}{Seções}
2744 \zcDicTypeTransl{section}{name-pl}{seções}
2745
2746 \zcDicTypeTransl{paragraph}{Name-sg}{Parágrafo}
2747 \zcDicTypeTransl{paragraph}{name-sg}{parágrafo}
2748 \zcDicTypeTransl{paragraph}{Name-pl}{Parágrafos}
2749 \zcDicTypeTransl{paragraph}{name-pl}{parágrafos}
2750 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}
2751 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}
2752 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}
2753 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}
2754

```



2755 \zcDicTypeTransl{appendix}{Name-sg}{Apêndice}  
 2756 \zcDicTypeTransl{appendix}{name-sg}{apêndice}  
 2757 \zcDicTypeTransl{appendix}{Name-pl}{Apêndices}  
 2758 \zcDicTypeTransl{appendix}{name-pl}{apêndices}  
 2759  
 2760 \zcDicTypeTransl{page}{Name-sg}{Página}  
 2761 \zcDicTypeTransl{page}{name-sg}{página}  
 2762 \zcDicTypeTransl{page}{Name-pl}{Páginas}  
 2763 \zcDicTypeTransl{page}{name-pl}{páginas}  
 2764 \zcDicTypeTransl{page}{name-sg-ab}{p.}  
 2765 \zcDicTypeTransl{page}{name-pl-ab}{pp.}  
 2766  
 2767 \zcDicTypeTransl{line}{Name-sg}{Linha}  
 2768 \zcDicTypeTransl{line}{name-sg}{linha}  
 2769 \zcDicTypeTransl{line}{Name-pl}{Linhas}  
 2770 \zcDicTypeTransl{line}{name-pl}{linhas}  
 2771  
 2772 \zcDicTypeTransl{figure}{Name-sg}{Figura}  
 2773 \zcDicTypeTransl{figure}{name-sg}{figura}  
 2774 \zcDicTypeTransl{figure}{Name-pl}{Figuras}  
 2775 \zcDicTypeTransl{figure}{name-pl}{figuras}  
 2776 \zcDicTypeTransl{figure}{Name-sg-ab}{Fig.}  
 2777 \zcDicTypeTransl{figure}{name-sg-ab}{fig.}  
 2778 \zcDicTypeTransl{figure}{Name-pl-ab}{Figs.}  
 2779 \zcDicTypeTransl{figure}{name-pl-ab}{figs.}  
 2780  
 2781 \zcDicTypeTransl{table}{Name-sg}{Tabela}  
 2782 \zcDicTypeTransl{table}{name-sg}{tabela}  
 2783 \zcDicTypeTransl{table}{Name-pl}{Tabelas}  
 2784 \zcDicTypeTransl{table}{name-pl}{tabelas}  
 2785  
 2786 \zcDicTypeTransl{item}{Name-sg}{Item}  
 2787 \zcDicTypeTransl{item}{name-sg}{item}  
 2788 \zcDicTypeTransl{item}{Name-pl}{Itens}  
 2789 \zcDicTypeTransl{item}{name-pl}{itens}  
 2790  
 2791 \zcDicTypeTransl{footnote}{Name-sg}{Nota}  
 2792 \zcDicTypeTransl{footnote}{name-sg}{nota}  
 2793 \zcDicTypeTransl{footnote}{Name-pl}{Notas}  
 2794 \zcDicTypeTransl{footnote}{name-pl}{notas}  
 2795  
 2796 \zcDicTypeTransl{note}{Name-sg}{Nota}  
 2797 \zcDicTypeTransl{note}{name-sg}{nota}  
 2798 \zcDicTypeTransl{note}{Name-pl}{Notas}  
 2799 \zcDicTypeTransl{note}{name-pl}{notas}  
 2800  
 2801 \zcDicTypeTransl{equation}{Name-sg}{Equação}  
 2802 \zcDicTypeTransl{equation}{name-sg}{equação}  
 2803 \zcDicTypeTransl{equation}{Name-pl}{Equações}  
 2804 \zcDicTypeTransl{equation}{name-pl}{equações}  
 2805 \zcDicTypeTransl{equation}{Name-sg-ab}{Eq.}  
 2806 \zcDicTypeTransl{equation}{name-sg-ab}{eq.}  
 2807 \zcDicTypeTransl{equation}{Name-pl-ab}{Eqs.}  
 2808 \zcDicTypeTransl{equation}{name-pl-ab}{eqs.}

2809 \zcDicTypeTransl{equation}{refpre-in}{()  
 2810 \zcDicTypeTransl{equation}{refpos-in}{()  
 2811  
 2812 \zcDicTypeTransl{theorem}{Name-sg}{Teorema}  
 2813 \zcDicTypeTransl{theorem}{name-sg}{teorema}  
 2814 \zcDicTypeTransl{theorem}{Name-pl}{Teoremas}  
 2815 \zcDicTypeTransl{theorem}{name-pl}{teoremas}  
 2816  
 2817 \zcDicTypeTransl{lemma}{Name-sg}{Lema}  
 2818 \zcDicTypeTransl{lemma}{name-sg}{lema}  
 2819 \zcDicTypeTransl{lemma}{Name-pl}{Lemas}  
 2820 \zcDicTypeTransl{lemma}{name-pl}{lemas}  
 2821  
 2822 \zcDicTypeTransl{corollary}{Name-sg}{Corolário}  
 2823 \zcDicTypeTransl{corollary}{name-sg}{corolário}  
 2824 \zcDicTypeTransl{corollary}{Name-pl}{Corolários}  
 2825 \zcDicTypeTransl{corollary}{name-pl}{corolários}  
 2826  
 2827 \zcDicTypeTransl{proposition}{Name-sg}{Proposição}  
 2828 \zcDicTypeTransl{proposition}{name-sg}{proposição}  
 2829 \zcDicTypeTransl{proposition}{Name-pl}{Proposições}  
 2830 \zcDicTypeTransl{proposition}{name-pl}{proposições}  
 2831  
 2832 \zcDicTypeTransl{definition}{Name-sg}{Definição}  
 2833 \zcDicTypeTransl{definition}{name-sg}{definição}  
 2834 \zcDicTypeTransl{definition}{Name-pl}{Definições}  
 2835 \zcDicTypeTransl{definition}{name-pl}{definições}  
 2836  
 2837 \zcDicTypeTransl{proof}{Name-sg}{Demonstração}  
 2838 \zcDicTypeTransl{proof}{name-sg}{demonstração}  
 2839 \zcDicTypeTransl{proof}{Name-pl}{Demonstrações}  
 2840 \zcDicTypeTransl{proof}{name-pl}{demonstrações}  
 2841  
 2842 \zcDicTypeTransl{result}{Name-sg}{Resultado}  
 2843 \zcDicTypeTransl{result}{name-sg}{resultado}  
 2844 \zcDicTypeTransl{result}{Name-pl}{Resultados}  
 2845 \zcDicTypeTransl{result}{name-pl}{resultados}  
 2846  
 2847 \zcDicTypeTransl{example}{Name-sg}{Exemplo}  
 2848 \zcDicTypeTransl{example}{name-sg}{exemplo}  
 2849 \zcDicTypeTransl{example}{Name-pl}{Exemplos}  
 2850 \zcDicTypeTransl{example}{name-pl}{exemplos}  
 2851  
 2852 \zcDicTypeTransl{remark}{Name-sg}{Observação}  
 2853 \zcDicTypeTransl{remark}{name-sg}{observação}  
 2854 \zcDicTypeTransl{remark}{Name-pl}{Observações}  
 2855 \zcDicTypeTransl{remark}{name-pl}{observações}  
 2856  
 2857 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}  
 2858 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}  
 2859 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}  
 2860 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}  
 2861  
 2862 \zcDicTypeTransl{listing}{Name-sg}{Listagem}

```

2863 \zcDicTypeTransl{listing}{name-sg}{listagem}
2864 \zcDicTypeTransl{listing}{Name-pl}{Listagens}
2865 \zcDicTypeTransl{listing}{name-pl}{listagens}
2866
2867 \zcDicTypeTransl{exercise}{Name-sg}{Exercício}
2868 \zcDicTypeTransl{exercise}{name-sg}{exercício}
2869 \zcDicTypeTransl{exercise}{Name-pl}{Exercícios}
2870 \zcDicTypeTransl{exercise}{name-pl}{exercícios}
2871
2872 \zcDicTypeTransl{solution}{Name-sg}{Solução}
2873 \zcDicTypeTransl{solution}{name-sg}{solução}
2874 \zcDicTypeTransl{solution}{Name-pl}{Soluções}
2875 \zcDicTypeTransl{solution}{name-pl}{soluções}
2876 \
```

## Spanish

```

2878 \ProvideDictionaryFor{Spanish}{zref-clever}
2879
2880 \zcDicDefaultTransl{namesep}{\nobreakspace}
2881 \zcDicDefaultTransl{pairsep}{~y\nobreakspace}
2882 \zcDicDefaultTransl{listsep}{~,~}
2883 \zcDicDefaultTransl{lastsep}{~y\nobreakspace}
2884 \zcDicDefaultTransl{tpairsep}{~y\nobreakspace}
2885 \zcDicDefaultTransl{tlistsep}{~,~}
2886 \zcDicDefaultTransl{tlastsep}{~y\nobreakspace}
2887 \zcDicDefaultTransl{notesep}{~}
2888 \zcDicDefaultTransl{rangesep}{~a\nobreakspace}
2889
2890 \zcDicTypeTransl{part}{Name-sg}{Parte}
2891 \zcDicTypeTransl{part}{name-sg}{parte}
2892 \zcDicTypeTransl{part}{Name-pl}{Partes}
2893 \zcDicTypeTransl{part}{name-pl}{partes}
2894
2895 \zcDicTypeTransl{chapter}{Name-sg}{Capítulo}
2896 \zcDicTypeTransl{chapter}{name-sg}{capítulo}
2897 \zcDicTypeTransl{chapter}{Name-pl}{Capítulos}
2898 \zcDicTypeTransl{chapter}{name-pl}{capítulos}
2899
2900 \zcDicTypeTransl{section}{Name-sg}{Sección}
2901 \zcDicTypeTransl{section}{name-sg}{sección}
2902 \zcDicTypeTransl{section}{Name-pl}{Secciones}
2903 \zcDicTypeTransl{section}{name-pl}{secciones}
2904
2905 \zcDicTypeTransl{paragraph}{Name-sg}{Párrafo}
2906 \zcDicTypeTransl{paragraph}{name-sg}{párrafo}
2907 \zcDicTypeTransl{paragraph}{Name-pl}{Párrafos}
2908 \zcDicTypeTransl{paragraph}{name-pl}{párrafos}
2909
2910 \zcDicTypeTransl{appendix}{Name-sg}{Apéndice}
2911 \zcDicTypeTransl{appendix}{name-sg}{apéndice}
2912 \zcDicTypeTransl{appendix}{Name-pl}{Apéndices}
2913 \zcDicTypeTransl{appendix}{name-pl}{apéndices}

```

2914  
 2915 \zcDicTypeTransl{page}{Name-sg}{Página}  
 2916 \zcDicTypeTransl{page}{name-sg}{página}  
 2917 \zcDicTypeTransl{page}{Name-pl}{Páginas}  
 2918 \zcDicTypeTransl{page}{name-pl}{páginas}  
 2919  
 2920 \zcDicTypeTransl{line}{Name-sg}{Línea}  
 2921 \zcDicTypeTransl{line}{name-sg}{línea}  
 2922 \zcDicTypeTransl{line}{Name-pl}{Líneas}  
 2923 \zcDicTypeTransl{line}{name-pl}{líneas}  
 2924  
 2925 \zcDicTypeTransl{figure}{Name-sg}{Figura}  
 2926 \zcDicTypeTransl{figure}{name-sg}{figura}  
 2927 \zcDicTypeTransl{figure}{Name-pl}{Figuras}  
 2928 \zcDicTypeTransl{figure}{name-pl}{figuras}  
 2929  
 2930 \zcDicTypeTransl{table}{Name-sg}{Cuadro}  
 2931 \zcDicTypeTransl{table}{name-sg}{cuadro}  
 2932 \zcDicTypeTransl{table}{Name-pl}{Cuadros}  
 2933 \zcDicTypeTransl{table}{name-pl}{cuadros}  
 2934  
 2935 \zcDicTypeTransl{item}{Name-sg}{Punto}  
 2936 \zcDicTypeTransl{item}{name-sg}{punto}  
 2937 \zcDicTypeTransl{item}{Name-pl}{Puntos}  
 2938 \zcDicTypeTransl{item}{name-pl}{puntos}  
 2939  
 2940 \zcDicTypeTransl{footnote}{Name-sg}{Nota}  
 2941 \zcDicTypeTransl{footnote}{name-sg}{nota}  
 2942 \zcDicTypeTransl{footnote}{Name-pl}{Notas}  
 2943 \zcDicTypeTransl{footnote}{name-pl}{notas}  
 2944  
 2945 \zcDicTypeTransl{note}{Name-sg}{Nota}  
 2946 \zcDicTypeTransl{note}{name-sg}{nota}  
 2947 \zcDicTypeTransl{note}{Name-pl}{Notas}  
 2948 \zcDicTypeTransl{note}{name-pl}{notas}  
 2949  
 2950 \zcDicTypeTransl{equation}{Name-sg}{Ecuación}  
 2951 \zcDicTypeTransl{equation}{name-sg}{ecuación}  
 2952 \zcDicTypeTransl{equation}{Name-pl}{Ecuaciones}  
 2953 \zcDicTypeTransl{equation}{name-pl}{ecuaciones}  
 2954 \zcDicTypeTransl{equation}{refpre-in}{(}  
 2955 \zcDicTypeTransl{equation}{refpos-in}{)}  
 2956  
 2957 \zcDicTypeTransl{theorem}{Name-sg}{Teorema}  
 2958 \zcDicTypeTransl{theorem}{name-sg}{teorema}  
 2959 \zcDicTypeTransl{theorem}{Name-pl}{Teoremas}  
 2960 \zcDicTypeTransl{theorem}{name-pl}{teoremas}  
 2961  
 2962 \zcDicTypeTransl{lemma}{Name-sg}{Lema}  
 2963 \zcDicTypeTransl{lemma}{name-sg}{lema}  
 2964 \zcDicTypeTransl{lemma}{Name-pl}{Lemas}  
 2965 \zcDicTypeTransl{lemma}{name-pl}{lemas}  
 2966  
 2967 \zcDicTypeTransl{corollary}{Name-sg}{Corolario}

2968 \zcDicTypeTransl{corollary}{name-sg}{corolario}  
 2969 \zcDicTypeTransl{corollary}{Name-pl}{Corolarios}  
 2970 \zcDicTypeTransl{corollary}{name-pl}{corolarios}  
 2971  
 2972 \zcDicTypeTransl{proposition}{Name-sg}{Proposición}  
 2973 \zcDicTypeTransl{proposition}{name-sg}{proposición}  
 2974 \zcDicTypeTransl{proposition}{Name-pl}{Proposiciones}  
 2975 \zcDicTypeTransl{proposition}{name-pl}{proposiciones}  
 2976  
 2977 \zcDicTypeTransl{definition}{Name-sg}{Definición}  
 2978 \zcDicTypeTransl{definition}{name-sg}{definición}  
 2979 \zcDicTypeTransl{definition}{Name-pl}{Definiciones}  
 2980 \zcDicTypeTransl{definition}{name-pl}{definiciones}  
 2981  
 2982 \zcDicTypeTransl{proof}{Name-sg}{Demostración}  
 2983 \zcDicTypeTransl{proof}{name-sg}{demostración}  
 2984 \zcDicTypeTransl{proof}{Name-pl}{Demostraciones}  
 2985 \zcDicTypeTransl{proof}{name-pl}{demostraciones}  
 2986  
 2987 \zcDicTypeTransl{result}{Name-sg}{Resultado}  
 2988 \zcDicTypeTransl{result}{name-sg}{resultado}  
 2989 \zcDicTypeTransl{result}{Name-pl}{Resultados}  
 2990 \zcDicTypeTransl{result}{name-pl}{resultados}  
 2991  
 2992 \zcDicTypeTransl{example}{Name-sg}{Ejemplo}  
 2993 \zcDicTypeTransl{example}{name-sg}{ejemplo}  
 2994 \zcDicTypeTransl{example}{Name-pl}{Ejemplos}  
 2995 \zcDicTypeTransl{example}{name-pl}{ejemplos}  
 2996  
 2997 \zcDicTypeTransl{remark}{Name-sg}{Observación}  
 2998 \zcDicTypeTransl{remark}{name-sg}{observación}  
 2999 \zcDicTypeTransl{remark}{Name-pl}{Observaciones}  
 3000 \zcDicTypeTransl{remark}{name-pl}{observaciones}  
 3001  
 3002 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}  
 3003 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}  
 3004 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}  
 3005 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}  
 3006  
 3007 \zcDicTypeTransl{listing}{Name-sg}{Listado}  
 3008 \zcDicTypeTransl{listing}{name-sg}{listado}  
 3009 \zcDicTypeTransl{listing}{Name-pl}{Listados}  
 3010 \zcDicTypeTransl{listing}{name-pl}{listados}  
 3011  
 3012 \zcDicTypeTransl{exercise}{Name-sg}{Ejercicio}  
 3013 \zcDicTypeTransl{exercise}{name-sg}{ejercicio}  
 3014 \zcDicTypeTransl{exercise}{Name-pl}{Ejercicios}  
 3015 \zcDicTypeTransl{exercise}{name-pl}{ejercicios}  
 3016  
 3017 \zcDicTypeTransl{solution}{Name-sg}{Solución}  
 3018 \zcDicTypeTransl{solution}{name-sg}{solución}  
 3019 \zcDicTypeTransl{solution}{Name-pl}{Soluciones}  
 3020 \zcDicTypeTransl{solution}{name-pl}{soluciones}  
 3021 </lang-spanish>

# Index

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

|  |                                   |
|--|-----------------------------------|
| <b>A</b>                                   |                                   |
| \AddToHook .....                           |                                   |
| . 91, 262, 282, 399, 472, 508, 534, 600    |                                   |
| \appendix .....                            | 54                                |
| \appendixname .....                        | 54                                |
| \Arg .....                                 | 2128                              |
| \AtEndOfPackage .....                      | 17, 506                           |
| <b>B</b>                                   |                                   |
| \baselanguage .....                        | 497, 558, 574                     |
| bool commands:                             |                                   |
| \bool_case_true: .....                     | 2                                 |
| \bool_if:NTF .....                         |                                   |
| .. 403, 407, 891, 924, 1291, 1313,         |                                   |
| 1386, 1511, 1532, 1563, 1618, 1685,        |                                   |
| 1689, 1696, 1705, 1711, 1992, 2196         |                                   |
| \bool_if:nTF ... 59, 953, 962, 971,        |                                   |
| 1040, 1068, 1091, 1180, 1188, 1327,        |                                   |
| 1335, 1544, 1551, 1558, 1807, 2070         |                                   |
| \bool_lazy_all:nTF .....                   | 2135                              |
| \bool_lazy_and:nnTF .....                  |                                   |
| ..... 848, 861, 1660, 1869, 2180           |                                   |
| \bool_lazy_any:nTF .....                   | 1960, 1969                        |
| \bool_lazy_or:nnTF .....                   | 852, 1857                         |
| \bool_new:N .....                          | 246,                              |
| 307, 308, 333, 360, 369, 376, 377,         |                                   |
| 432, 433, 450, 451, 593, 594, 840,         |                                   |
| 944, 1220, 1221, 1232, 1233, 1234, 1254    |                                   |
| \bool_set:Nn .....                         | 846                               |
| \bool_set_false:N .. 253, 267, 272,        |                                   |
| 291, 302, 320, 324, 384, 393, 394,         |                                   |
| 409, 615, 1029, 1280, 1319, 1333,          |                                   |
| 1344, 1523, 1658, 1659, 1967, 1984         |                                   |
| \bool_set_true:N .....                     |                                   |
| . 258, 314, 315, 319, 325, 383, 388,       |                                   |
| 389, 604, 609, 1052, 1063, 1080,           |                                   |
| 1086, 1103, 1109, 1135, 1147, 1287,        |                                   |
| 1314, 1320, 1324, 1345, 1348, 1983,        |                                   |
| 2206, 2213, 2214, 2232, 2239, 2240         |                                   |
| \bool_until_do:Nn .....                    | 1033, 1281                        |
| <b>C</b>                                   |                                   |
| clist commands:                            |                                   |
| \clist_map_inline:Nn .....                 | 544, 567                          |
| \clist_map_inline:nn .....                 |                                   |
| .... 200, 627, 673, 689, 753, 778, 808     |                                   |
| \counterwithin .....                       | 4                                 |
| \cs .....                                  | 537, 1159, 1509, 2131, 2165, 2176 |
| cs commands:                               |                                   |
| \cs_generate_variant:Nn .....              |                                   |
| ..... 55, 56, 149, 152, 945, 1848          |                                   |
| \cs_if_exist:NTF .....                     | 39, 48, 69                        |
| \cs_new:Npn 37, 46, 57, 67, 78, 1803, 1987 |                                   |
| \cs_new_protected:Npn .....                |                                   |
| .... 147, 150, 153, 161, 841, 888,         |                                   |
| 931, 946, 1010, 1155, 1211, 1257,          |                                   |
| 1397, 1653, 1849, 2129, 2174, 2194         |                                   |
| \cs_new_protected:Npx .....                | 90                                |
| \cs_set_eq:NN .....                        | 94                                |
| <b>D</b>                                   |                                   |
| \declaretranslation .....                  | 151, 154                          |
| \dots .....                                | 973, 983, 987                     |
| <b>E</b>                                   |                                   |
| \endinput .....                            | 12                                |
| exp commands:                              |                                   |
| \exp_args:NNe .....                        | 27                                |
| \exp_args:NNx .....                        | 95, 1077, 1100                    |
| \exp_args:Nx 492, 496, 553, 557, 569, 573  |                                   |
| \exp_args:Nxx .....                        |                                   |
| ..... 994, 1048, 2198, 2220, 2224          |                                   |
| \exp_not:N .....                           | 103,                              |
| 109, 1565, 1568, 1588, 1591, 1594,         |                                   |
| 1810, 1813, 1816, 1828, 1830, 1833,        |                                   |
| 1836, 1841, 1843, 1846, 1990, 1998,        |                                   |
| 2016, 2019, 2021, 2024, 2030, 2037,        |                                   |
| 2039, 2043, 2046, 2049, 2051, 2057,        |                                   |
| 2061, 2064, 2076, 2079, 2082, 2105,        |                                   |
| 2107, 2110, 2113, 2119, 2121, 2124         |                                   |
| \exp_not:n . 1417, 1433, 1445, 1449,       |                                   |
| 1469, 1482, 1485, 1497, 1500, 1533,        |                                   |
| 1534, 1566, 1587, 1592, 1593, 1725,        |                                   |
| 1738, 1743, 1763, 1774, 1777, 1787,        |                                   |
| 1790, 1811, 1812, 1814, 1824, 1826,        |                                   |
| 1829, 1834, 1835, 1837, 1838, 1840,        |                                   |
| 1842, 2017, 2018, 2020, 2022, 2023,        |                                   |
| 2025, 2026, 2029, 2041, 2042, 2047,        |                                   |
| 2048, 2050, 2058, 2062, 2063, 2065,        |                                   |
| 2077, 2078, 2080, 2100, 2103, 2106,        |                                   |
| 2111, 2112, 2114, 2115, 2118, 2120         |                                   |
| <b>F</b>                                   |                                   |
| file commands:                             |                                   |
| \file_if_exist:nTF .....                   |                                   |
| ..... 492, 496, 553, 557, 569, 573         |                                   |
| \fmtversion .....                          | 3                                 |

|                    |  |                                       |  |
|--------------------|--|---------------------------------------|--|
| <b>G</b>           |  | <b>M</b>                              |  |
| group commands:    |  | \MessageBreak                         | 10   |
| \group_begin:      | 93, 843, 1565, 1591, 1810, 1813, 1833, 1836, 2016, 2021, 2024, 2039, 2046, 2061, 2076, 2079, 2110, 2113  | \meta                                 | 1197   |
| \group_end:        | 96, 869, 1588, 1594, 1828, 1830, 1841, 1843, 2019, 2030, 2037, 2043, 2049, 2064, 2105, 2107, 2119, 2121  | msg commands:                         |  |
|                    |  | \msg_line_context:                    | 102, 108, 115, 129, 133, 135, 137, 139   |
|                    |  | \msg_new:nnn                          | 100, 106, 111, 113, 118, 123, 125, 127, 132, 134, 136, 138   |
|                    |  | \msg_warning:nn                       | 270, 300, 408, 414, 598, 619   |
|                    |  | \msg_warning:nnn                      | 228, 685, 772, 827, 1361, 1518, 1904, 1951   |
|                    |  | \msg_warning:nnnn                     | 174, 1061, 1084, 1107, 1145  |
| <b>H</b>           |  |                                       |  |
| \hyperlink         | 45   |                                       |  |
| <b>I</b>           |  | <b>N</b>                              |  |
| \IfBooleanTF       | 873  | \newcounter                           | 4  |
| \IfFormatAtLeastTF | 3, 4   | \NewDocumentCommand                   | 155, 157, 662, 666, 734, 837, 871  |
| \IfTranslation     | 142  | \NewHook                              | 467  |
| int commands:      |  | \newtheorem                           | 55   |
| \int_case:nnTF     | 1400, 1426, 1457, 1621, 1718, 1754   | \nobreakspace                         | 2247, 2264, 2265, 2267, 2268, 2270, 2272, 2427, 2428, 2430, 2431, 2433, 2435, 2576, 2577, 2579, 2580, 2582, 2584, 2721, 2722, 2724, 2725, 2727, 2729, 2880, 2881, 2883, 2884, 2886, 2888 |
| \int_compare:nNnTF | 998, 1053, 1120, 1136, 1166, 1168, 1213, 1368, 1413, 1447, 1610, 1612, 1675, 1699, 1741, 2202, 2208, 2228, 2234  | \noexpand                             | 121  |
| \int_compare_p:nNn | 1182, 1190, 1861, 1872, 1980   |                                       |  |
| \int_eval:n        | 90   |                                       |  |
| \int_incr:N        | 1648, 1688, 1690, 1704, 1706, 1710, 1712, 1801   |                                       |  |
| \int_new:N         | 877, 878, 1227, 1228, 1229, 1230   |                                       |  |
| \int_set:Nn        | 1167, 1169, 1173, 1176   |                                       |  |
| \int_use:N         | 33, 35, 50   |                                       |  |
| \int_zero:N        | 1157, 1158, 1265, 1266, 1267, 1268, 1647, 1649, 1650, 1796, 1797   |                                       |  |
| iow commands:      |  |                                       |  |
| \iow_newline:      | 120, 124   |                                       |  |
| <b>K</b>           |  | <b>P</b>                              |  |
| keys commands:     |  | \PackageError                         | 7  |
| \keys_define:nn    | 21, 168, 196, 222, 247, 286, 296, 309, 334, 343, 361, 370, 378, 411, 418, 434, 452, 468, 510, 588, 595, 605, 616, 624, 649, 681, 715, 740, 761, 791, 820 | \pagenumbering                        | 6  |
| \keys_set:nn       | 21, 610, 663, 671, 738, 844  | \pkg                                  | 536, 546, 549  |
| keyval commands:   |  | prg commands:                         |  |
| \keyval_parse:nnn  | 172, 226   | \prg_generate_conditional_variant:Nnn | 146  |
|                    |  | \prg_new_conditional:Npnn             | 140  |
|                    |  | \prg_return_false:                    | 144  |
|                    |  | \prg_return_true:                     | 143  |
|                    |  | \ProcessKeysOptions                   | 17, 661  |
|                    |  | prop commands:                        |  |
|                    |  | \prop_get:NnN                         | 2148   |
|                    |  | \prop_get:NnNTF                       | 1883, 1910, 1915, 2132, 2177, 2187   |
|                    |  | \prop_if_exist:NTF                    | 668, 747   |
|                    |  | \prop_if_exist_p:N                    | 2139, 2183   |
|                    |  | \prop_if_in:NnTF                      | 25   |
|                    |  | \prop_if_in_p:Nn                      | 60, 2143   |
|                    |  | \prop_item:Nn                         | 27, 61   |
|                    |  | \prop_new:N                           | 167, 221, 626, 669, 748  |
|                    |  | \prop_put:Nnn                         | 165, 656, 727  |
|                    |  | \prop_remove:Nn                       | 164, 655, 722  |
|                    |  | \providecommand                       | 3  |
|                    |  | \ProvideDictionaryFor                 | 8, 2262, 2425, 2574, 2719, 2878  |
| <b>L</b>           |  |                                       |  |
| \labelformat       | 3  |                                       |  |
| \LoadDictionaryFor | 494, 498, 555, 559, 571, 575, 581  |                                       |  |







|                                     |  |
|-------------------------------------|--|
| 2648, 2649, 2650, 2651, 2653, 2654, |  |
| 2655, 2656, 2658, 2659, 2660, 2661, |  |
| 2663, 2664, 2665, 2666, 2668, 2669, |  |
| 2670, 2671, 2673, 2674, 2675, 2676, |  |
| 2678, 2679, 2680, 2681, 2683, 2684, |  |
| 2685, 2686, 2688, 2689, 2690, 2691, |  |
| 2693, 2694, 2695, 2696, 2698, 2699, |  |
| 2700, 2701, 2703, 2704, 2705, 2706, |  |
| 2708, 2709, 2710, 2711, 2713, 2714, |  |
| 2715, 2716, 2731, 2732, 2733, 2734, |  |
| 2736, 2737, 2738, 2739, 2741, 2742, |  |
| 2743, 2744, 2746, 2747, 2748, 2749, |  |
| 2750, 2751, 2752, 2753, 2755, 2756, |  |
| 2757, 2758, 2760, 2761, 2762, 2763, |  |
| 2764, 2765, 2767, 2768, 2769, 2770, |  |
| 2772, 2773, 2774, 2775, 2776, 2777, |  |
| 2778, 2779, 2781, 2782, 2783, 2784, |  |
| 2786, 2787, 2788, 2789, 2791, 2792, |  |
| 2793, 2794, 2796, 2797, 2798, 2799, |  |
| 2801, 2802, 2803, 2804, 2805, 2806, |  |
| 2807, 2808, 2809, 2810, 2812, 2813, |  |
| 2814, 2815, 2817, 2818, 2819, 2820, |  |
| 2822, 2823, 2824, 2825, 2827, 2828, |  |
| 2829, 2830, 2832, 2833, 2834, 2835, |  |
| 2837, 2838, 2839, 2840, 2842, 2843, |  |
| 2844, 2845, 2847, 2848, 2849, 2850, |  |
| 2852, 2853, 2854, 2855, 2857, 2858, |  |
| 2859, 2860, 2862, 2863, 2864, 2865, |  |
| 2867, 2868, 2869, 2870, 2872, 2873, |  |
| 2874, 2875, 2890, 2891, 2892, 2893, |  |
| 2895, 2896, 2897, 2898, 2900, 2901, |  |
| 2902, 2903, 2905, 2906, 2907, 2908, |  |
| 2910, 2911, 2912, 2913, 2915, 2916, |  |
| 2917, 2918, 2920, 2921, 2922, 2923, |  |
| 2925, 2926, 2927, 2928, 2930, 2931, |  |
| 2932, 2933, 2935, 2936, 2937, 2938, |  |
| 2940, 2941, 2942, 2943, 2945, 2946, |  |
| 2947, 2948, 2950, 2951, 2952, 2953, |  |
| 2954, 2955, 2957, 2958, 2959, 2960, |  |
| 2962, 2963, 2964, 2965, 2967, 2968, |  |
| 2969, 2970, 2972, 2973, 2974, 2975, |  |
| 2977, 2978, 2979, 2980, 2982, 2983, |  |
| 2984, 2985, 2987, 2988, 2989, 2990, |  |
| 2992, 2993, 2994, 2995, 2997, 2998, |  |
| 2999, 3000, 3002, 3003, 3004, 3005, |  |
| 3007, 3008, 3009, 3010, 3012, 3013, |  |
| 3014, 3015, 3017, 3018, 3019, 3020  |  |
| \zcheck                             | 121  |
| \zcpageref                          | 25, 871  |
| \zceref                             | 24, 26, 33, 35, 837, 874, 875  |
| \zcRefTypeSetup                     | 20, 109, 666   |
| \zcsetup                            | 20, 662  |
| zrefcheck commands:                 |  |
| \zrefcheck_zceref_beg_label:        | 851  |
| \zrefcheck_zceref_end_label_-       |  |
| maybe:                              | 865  |
| \zrefcheck_zceref_run_checks_on_-   |  |
| labels:n                            | 866  |
| zrefclever internal commands:       |  |
| \l__zrefclever_abbrev_bool          | 450, 454, 1870   |
| \l__zrefclever_capitalize_bool      | 432, 436, 1858   |
| \l__zrefclever_capitalize_first_-   |  |
| bool                                | 433, 442, 1860   |
| \__zrefclever_counter_reset_by:n    | 5, 10, 39, 41, 43, 48, 50, 52, 57  |
| \__zrefclever_counter_reset_by_-    |  |
| aux:nn                              | 64, 67   |
| \__zrefclever_counter_reset_by_-    |  |
| auxi:nnn                            | 74, 78   |
| \l__zrefclever_counter_resetby_-    |  |
| prop                                | 4, 10, 60, 61, 221, 233  |
| \l__zrefclever_counter_resettters_- |  |
| seq                                 | 4, 10, 63, 195, 202, 205   |
| \l__zrefclever_counter_type_prop    | 3, 9, 25, 28, 167, 179   |
| \l__zrefclever_current_language_-   |  |
| tl                                  | 466, 485, 525, 538   |
| \__zrefclever_declare_default_-     |  |
| transl:nnn                          | 8, 153, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259                                   |
| \__zrefclever_declare_transl:nnn    | 8, 150, 768, 798, 802, 831   |
| \__zrefclever_get_enclosing_-       |  |
| counters:n                          | 4, 5, 37, 42, 84   |
| \__zrefclever_get_enclosing_-       |  |
| counters_value:n                    | 4, 5, 37, 51, 86   |
| \__zrefclever_get_option_-          |  |
| plain:nN                            | 1370, 1371, 1372, 2174   |
| \__zrefclever_get_option_with_-     |  |
| transl:nN                           | 55, 1271, 1272, 1273, 1274, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 2128                                 |
| \__zrefclever_get_ref:n             | 1418, 1434, 1446, 1450, 1470, 1483, 1486, 1498, 1501, 1535, 1555, 1726, 1739, 1744, 1764, 1775, 1778, 1788, 1791, 1803 |
| \__zrefclever_get_ref_first:        | 35, 45, 1548, 1604, 1987   |
| \__zrefclever_get_transl:nnn        | 8, 147, 1895, 1927, 1942, 2157, 2167   |
| \__zrefclever_if_transl:nn          | 140, 146   |
| \__zrefclever_if_transl:nnTF        | 7, 140, 1888, 1920, 1935, 2153   |
| \l__zrefclever_label_a_tl           | 879, 1283, 1300, 1316, 1355,   |

1356, 1362, 1406, 1418, 1434, 1450,  
 1486, 1501, 1528, 1535, 1666, 1670,  
 1679, 1703, 1726, 1744, 1778, 1791  
 \l\_zrefclever\_label\_b\_tl .....  
     879, 1286, 1289, 1305, 1318, 1323, 1670  
 \l\_zrefclever\_label\_count\_int ..  
     ..... 34, 1227,  
     1265, 1368, 1400, 1647, 1675, 1801  
 \l\_zrefclever\_label\_enclcnt\_a\_-  
     tl ..... 879, 1012,  
     1014, 1015, 1036, 1101, 1125, 1126  
 \l\_zrefclever\_label\_enclcnt\_b\_-  
     tl ..... 879, 1016,  
     1018, 1019, 1038, 1078, 1127, 1128  
 \l\_zrefclever\_label\_enclval\_a\_-  
     tl ..... 879, 1020,  
     1022, 1023, 1121, 1129, 1130, 1137  
 \l\_zrefclever\_label\_enclval\_b\_-  
     tl ..... 879, 1024,  
     1026, 1027, 1123, 1131, 1132, 1139  
 \l\_zrefclever\_label\_type\_a\_tl ..  
     ..... 879, 933, 935, 937,  
     940, 948, 957, 966, 974, 981, 1172,  
     1201, 1293, 1297, 1330, 1338, 1343,  
     1359, 1407, 1680, 2137, 2140, 2144,  
     2149, 2155, 2159, 2181, 2184, 2188  
 \l\_zrefclever\_label\_type\_b\_tl ..  
     879, 950, 958, 967, 975, 981, 1175,  
     1204, 1294, 1302, 1331, 1339, 1343  
 \\_zrefclever\_label\_type\_put\_-  
     new\_right:n ..... 27, 894, 931  
 \l\_zrefclever\_label\_types\_seq ..  
     ..... 27, 887, 890, 937, 940, 1199  
 \\_zrefclever\_labels\_in\_sequence:nn  
     ..... 1527, 1669, 2194  
 \l\_zrefclever\_last\_of\_type\_bool  
     ..... 33, 1220, 1314, 1319, 1320,  
     1324, 1333, 1344, 1345, 1348, 1386  
 \l\_zrefclever\_lastsep\_tl . 1243,  
     1377, 1433, 1449, 1469, 1485, 1497  
 \l\_zrefclever\_link\_star\_bool ...  
     ..... 839, 846, 1808, 1963, 2073  
 \l\_zrefclever\_listsep\_tl .....  
     ... 1242, 1376, 1445, 1482, 1725,  
     1738, 1743, 1763, 1774, 1777, 1787  
 \l\_zrefclever\_main\_language\_tl .  
     ..... 465, 479, 519, 543, 566, 580  
 \l\_zrefclever\_name\_format\_-  
     fallback\_tl ..... 1253,  
     1876, 1879, 1881, 1917, 1939, 1946  
 \l\_zrefclever\_name\_format\_tl ...  
     ..... 1253,  
     1863, 1864, 1867, 1868, 1876, 1877,  
     1885, 1892, 1899, 1912, 1924, 1931  
 \l\_zrefclever\_name\_in\_link\_bool  
     ... 1253, 1563, 1967, 1983, 1984, 1992  
 \l\_zrefclever\_namefont\_tl 1235,  
     1370, 1566, 1592, 2017, 2047, 2062  
 \l\_zrefclever\_nameinlink\_str ...  
     ..... 417, 422,  
     424, 426, 428, 1965, 1971, 1973, 1977  
 \l\_zrefclever\_nameinlink\_tl .. 417  
 \l\_zrefclever\_namesep\_tl .....  
     ... 1239, 1373, 2020, 2050, 2058, 2065  
 \l\_zrefclever\_next\_is\_same\_bool  
     ..... 34, 53, 1229,  
     1659, 1689, 1705, 1711, 2214, 2240  
 \l\_zrefclever\_next\_maybe\_range\_-  
     bool .....  
     ... 34, 53, 1229, 1523, 1532, 1658,  
     1685, 1696, 2206, 2213, 2232, 2239  
 \l\_zrefclever\_noabbrev\_first\_-  
     bool ..... 451, 460, 1873  
 \l\_zrefclever\_notesept\_tl .....  
     ..... 858, 1248, 1274  
 \\_zrefclever\_page\_format\_aux: ..  
     ..... 90, 94  
 \g\_zrefclever\_page\_format\_tl ...  
     ..... 6, 89, 95, 98  
 \l\_zrefclever\_page\_ref\_bool ...  
     ..... 246, 253, 258,  
     267, 272, 291, 302, 891, 924, 1291, 2196  
 \l\_zrefclever\_pairsep\_tl .....  
     ..... 1241, 1375, 1417, 1533  
 \\_zrefclever\_prop\_put\_non\_-  
     empty:Nnn ..... 9, 161, 178, 232  
 \l\_zrefclever\_range\_beg\_label\_-  
     tl ..... 34, 1229, 1264,  
     1446, 1465, 1470, 1480, 1483, 1495,  
     1498, 1646, 1687, 1703, 1736, 1739,  
     1761, 1764, 1772, 1775, 1785, 1788  
 \l\_zrefclever\_range\_count\_int ..  
     ..... 34,  
     1229, 1267, 1426, 1458, 1649, 1688,  
     1700, 1704, 1710, 1718, 1755, 1796  
 \l\_zrefclever\_range\_inhibit\_-  
     next\_bool ..... 33, 34, 1229, 1664  
 \l\_zrefclever\_range\_same\_count\_-  
     int ..... 34,  
     1229, 1268, 1413, 1447, 1458, 1650,  
     1690, 1706, 1712, 1741, 1755, 1797  
 \l\_zrefclever\_rangesep\_tl .....  
     ..... 1240, 1374, 1500, 1534, 1790  
 \l\_zrefclever\_ref\_language\_tl ..  
     ..... 464, 479, 485, 489,  
     519, 525, 528, 1889, 1896, 1921,  
     1928, 1936, 1943, 2154, 2158, 2168

|   |   |
|---|---|
| <code>\l__zrefclever_ref_options_prop .</code>  | <code>\l__zrefclever_type_first_label_-</code>  |
| ..... <a href="#">20</a> , <a href="#">626</a> , <a href="#">655</a> , <a href="#">656</a> , <a href="#">2132</a> , <a href="#">2177</a>                                      | tl ..... <a href="#">1222</a> , <a href="#">1262</a> , <a href="#">1406</a> , <a href="#">1515</a> ,  |
| <code>\l__zrefclever_ref_property_tl .</code>   | <a href="#">1524</a> , <a href="#">1528</a> , <a href="#">1555</a> , <a href="#">1571</a> , <a href="#">1574</a> , <a href="#">1579</a> ,                 |
| ..... <a href="#">12</a> , <a href="#">245</a> , <a href="#">252</a> ,  | <a href="#">1585</a> , <a href="#">1644</a> , <a href="#">1679</a> , <a href="#">1851</a> , <a href="#">1989</a> , <a href="#">1995</a> ,                 |
| <a href="#">257</a> , <a href="#">266</a> , <a href="#">271</a> , <a href="#">290</a> , <a href="#">301</a> , <a href="#">1805</a> , <a href="#">1825</a> ,                   | <a href="#">2001</a> , <a href="#">2003</a> , <a href="#">2007</a> , <a href="#">2012</a> , <a href="#">2027</a> , <a href="#">2068</a> ,                 |
| <a href="#">1839</a> , <a href="#">1995</a> , <a href="#">2028</a> , <a href="#">2068</a> , <a href="#">2102</a> , <a href="#">2117</a>                                       | <a href="#">2085</a> , <a href="#">2087</a> , <a href="#">2091</a> , <a href="#">2096</a> , <a href="#">2101</a> , <a href="#">2116</a>                   |
| <code>\l__zrefclever_ref_typeset_font_-</code>  | <code>\l__zrefclever_type_first_label_-</code>  |
| tl ..... <a href="#">623</a> , <a href="#">625</a> , <a href="#">1277</a>   | type_tl ..... <a href="#">1222</a> ,  |
| <code>\l__zrefclever_reffont_in_tl</code> <a href="#">1237</a> ,  | <a href="#">1263</a> , <a href="#">1407</a> , <a href="#">1519</a> , <a href="#">1645</a> , <a href="#">1680</a> , <a href="#">1854</a> ,                 |
| <a href="#">1372</a> , <a href="#">1814</a> , <a href="#">1837</a> , <a href="#">2025</a> , <a href="#">2080</a> , <a href="#">2114</a>                                       | <a href="#">1884</a> , <a href="#">1891</a> , <a href="#">1898</a> , <a href="#">1905</a> , <a href="#">1911</a> ,  |
| <code>\l__zrefclever_reffont_out_tl .</code>  | <a href="#">1916</a> , <a href="#">1923</a> , <a href="#">1930</a> , <a href="#">1938</a> , <a href="#">1945</a> , <a href="#">1952</a>                   |
| ..... <a href="#">1236</a> , <a href="#">1371</a> ,   | <code>\__zrefclever_type_name_setup: .</code>   |
| <a href="#">1811</a> , <a href="#">1834</a> , <a href="#">2022</a> , <a href="#">2041</a> , <a href="#">2077</a> , <a href="#">2111</a>                                       | ..... <a href="#">35</a> , <a href="#">1543</a> , <a href="#">1849</a>  |
| <code>\l__zrefclever_refpos_in_tl</code> <a href="#">1252</a> ,   | <code>\l__zrefclever_type_name_tl .</code> <a href="#">46</a> ,   |
| <a href="#">1381</a> , <a href="#">1826</a> , <a href="#">1840</a> , <a href="#">2029</a> , <a href="#">2103</a> , <a href="#">2118</a>                                       | <a href="#">1253</a> , <a href="#">1587</a> , <a href="#">1593</a> , <a href="#">1852</a> , <a href="#">1855</a> , <a href="#">1886</a> ,                 |
| <code>\l__zrefclever_refpos_out_tl</code> <a href="#">1250</a> ,  | <a href="#">1895</a> , <a href="#">1903</a> , <a href="#">1913</a> , <a href="#">1918</a> , <a href="#">1927</a> , <a href="#">1942</a> ,                 |
| <a href="#">1379</a> , <a href="#">1829</a> , <a href="#">1842</a> , <a href="#">2042</a> , <a href="#">2106</a> , <a href="#">2120</a>                                       | <a href="#">1950</a> , <a href="#">1964</a> , <a href="#">2018</a> , <a href="#">2048</a> , <a href="#">2055</a> , <a href="#">2063</a>                   |
| <code>\l__zrefclever_refpre_in_tl</code> <a href="#">1251</a> ,   | <code>\l__zrefclever_typeset_compress_-</code>  |
| <a href="#">1380</a> , <a href="#">1824</a> , <a href="#">1838</a> , <a href="#">2026</a> , <a href="#">2100</a> , <a href="#">2115</a>                                       | bool ..... <a href="#">360</a> , <a href="#">363</a> , <a href="#">1661</a>   |
| <code>\l__zrefclever_refpre_out_tl</code> <a href="#">1249</a> ,  | <code>\l__zrefclever_typeset_labels_-</code>  |
| <a href="#">1378</a> , <a href="#">1812</a> , <a href="#">1835</a> , <a href="#">2023</a> , <a href="#">2078</a> , <a href="#">2112</a>                                       | seq ... <a href="#">1222</a> , <a href="#">1259</a> , <a href="#">1283</a> , <a href="#">1284</a> , <a href="#">1289</a>                                  |
| <code>\l__zrefclever_setup_language_tl</code>   | <code>\l__zrefclever_typeset_last_bool</code>   |
| ..... <a href="#">664</a> , <a href="#">736</a> , <a href="#">768</a> , <a href="#">798</a> , <a href="#">802</a> , <a href="#">831</a>                                       | ..... <a href="#">33</a> , <a href="#">1220</a> ,   |
| <code>\l__zrefclever_setup_type_tl .</code>   | <a href="#">1280</a> , <a href="#">1281</a> , <a href="#">1287</a> , <a href="#">1313</a> , <a href="#">1618</a> , <a href="#">1979</a>                   |
| ..... <a href="#">664</a> , <a href="#">670</a> , <a href="#">723</a> , <a href="#">728</a> ,   | <code>\l__zrefclever_typeset_name_bool</code>   |
| <a href="#">737</a> , <a href="#">745</a> , <a href="#">749</a> , <a href="#">766</a> , <a href="#">796</a> , <a href="#">803</a> , <a href="#">825</a> , <a href="#">832</a> | ..... <a href="#">308</a> , <a href="#">315</a> , <a href="#">320</a> , <a href="#">325</a> , <a href="#">1545</a> , <a href="#">1559</a>                 |
| <code>\l__zrefclever_sort_decided_bool</code>   | <code>\l__zrefclever_typeset_queue_-</code>   |
| .... <a href="#">944</a> , <a href="#">1029</a> , <a href="#">1033</a> , <a href="#">1052</a> , <a href="#">1063</a> ,  | curr_tl ..... <a href="#">1222</a> , <a href="#">1261</a> , <a href="#">1415</a> ,  |
| <a href="#">1080</a> , <a href="#">1086</a> , <a href="#">1103</a> , <a href="#">1109</a> , <a href="#">1135</a> , <a href="#">1147</a>                                       | <a href="#">1431</a> , <a href="#">1440</a> , <a href="#">1467</a> , <a href="#">1477</a> , <a href="#">1492</a> , <a href="#">1513</a> ,                 |
| <code>\__zrefclever_sort_default:nn .</code>  | <a href="#">1530</a> , <a href="#">1547</a> , <a href="#">1554</a> , <a href="#">1561</a> , <a href="#">1604</a> , <a href="#">1625</a> ,                 |
| ..... <a href="#">26</a> , <a href="#">27</a> , <a href="#">926</a> , <a href="#">946</a>   | <a href="#">1630</a> , <a href="#">1636</a> , <a href="#">1642</a> , <a href="#">1643</a> , <a href="#">1723</a> , <a href="#">1734</a> ,                 |
| <code>\__zrefclever_sort_default_-</code>   | <a href="#">1759</a> , <a href="#">1770</a> , <a href="#">1783</a> , <a href="#">1866</a> , <a href="#">1974</a> , <a href="#">1978</a>                   |
| different_types:nn .... <a href="#">988</a> , <a href="#">1155</a>  | <code>\l__zrefclever_typeset_queue_-</code>   |
| <code>\__zrefclever_sort_default_same_-</code>  | prev_tl .... <a href="#">1222</a> , <a href="#">1260</a> , <a href="#">1614</a> , <a href="#">1642</a>  |
| type:nn ..... <a href="#">984</a> , <a href="#">1010</a>  | <code>\l__zrefclever_typeset_range_-</code>   |
| <code>\__zrefclever_sort_labels: .</code>   | bool ..... <a href="#">369</a> , <a href="#">372</a> , <a href="#">854</a> , <a href="#">1511</a>   |
| ..... <a href="#">27</a> , <a href="#">32</a> , <a href="#">855</a> , <a href="#">888</a>   | <code>\l__zrefclever_typeset_ref_bool .</code>  |
| <code>\__zrefclever_sort_page:nn .</code>   | ..... <a href="#">307</a> , <a href="#">314</a> , <a href="#">319</a> , <a href="#">324</a> , <a href="#">1545</a> , <a href="#">1552</a>                 |
| ..... <a href="#">33</a> , <a href="#">925</a> , <a href="#">1211</a>   | <code>\__zrefclever_typeset_refs: .</code>  |
| <code>\l__zrefclever_sort_prior_a_int .</code>  | ..... <a href="#">33</a> , <a href="#">34</a> , <a href="#">45</a> , <a href="#">46</a> , <a href="#">49</a> , <a href="#">856</a> , <a href="#">1257</a> |
| <a href="#">877</a> , <a href="#">1157</a> , <a href="#">1166</a> , <a href="#">1167</a> , <a href="#">1173</a> , <a href="#">1183</a> , <a href="#">1191</a>                 | <code>\__zrefclever_typeset_refs_aux_-</code>   |
| <code>\l__zrefclever_sort_prior_b_int .</code>  | last_of_type: ..... <a href="#">1389</a> , <a href="#">1397</a>   |
| <a href="#">878</a> , <a href="#">1158</a> , <a href="#">1168</a> , <a href="#">1169</a> , <a href="#">1176</a> , <a href="#">1184</a> , <a href="#">1192</a>                 | <code>\__zrefclever_typeset_refs_aux_-</code>   |
| <code>\l__zrefclever_tlastsep_tl .</code>   | not_last_of_type: .... <a href="#">1393</a> , <a href="#">1653</a>  |
| ..... <a href="#">1247</a> , <a href="#">1273</a> , <a href="#">1635</a>  | <code>\l__zrefclever_typeset_sort_bool</code>   |
| <code>\l__zrefclever_tlistsep_tl .</code>   | ..... <a href="#">333</a> , <a href="#">336</a> , <a href="#">853</a>   |
| ..... <a href="#">1246</a> , <a href="#">1272</a> , <a href="#">1613</a>  | <code>\l__zrefclever_typesort_seq .</code>  |
| <code>\l__zrefclever_tpairsep_tl .</code>   | ..... <a href="#">342</a> , <a href="#">347</a> , <a href="#">351</a> , <a href="#">357</a> , <a href="#">1162</a>  |
| ..... <a href="#">1245</a> , <a href="#">1271</a> , <a href="#">1629</a>  | <code>\l__zrefclever_use_hyperref_bool</code>   |
| <code>\l__zrefclever_type&lt;type&gt;-</code>   | ..... <a href="#">376</a> , <a href="#">403</a> , <a href="#">409</a> , <a href="#">1808</a> , <a href="#">1962</a> , <a href="#">2072</a>                |
| options_prop ..... <a href="#">20</a>   | <code>\l__zrefclever_warn_hyperref_-</code>   |
| <code>\l__zrefclever_type_count_int .</code>  | bool ..... <a href="#">376</a> , <a href="#">407</a>  |
| ..... <a href="#">34</a> , <a href="#">1227</a> , <a href="#">1266</a> , <a href="#">1610</a> ,   | <code>\__zrefclever_zcref:nnn .</code> <a href="#">838</a> , <a href="#">841</a>  |
| <a href="#">1612</a> , <a href="#">1621</a> , <a href="#">1648</a> , <a href="#">1861</a> , <a href="#">1872</a> , <a href="#">1980</a>                                       | <code>\__zrefclever_zcref:nnnn .</code> <a href="#">24</a> , <a href="#">26</a> , <a href="#">841</a>   |

|   |                           |                         |
|---|---------------------------|-------------------------|
| \l_zrefclever_zcref_labels_seq .          | bool . . . . .            | 594, 609, 850, 863      |
| . . . . 26, 839, 845, 867, 894, 896, 1259 |                           |                         |
| \l_zrefclever_zcref_note_tl . . .         | \l_zrefclever_zrefcheck_- |                         |
| . . . . . 587, 590, 859                   | available_bool . . . . .  |                         |
| \l_zrefclever_zcref_with_check_-          | . . . . .                 | 593, 604, 615, 849, 862 |