

The zref-clever package implementation*

Gustavo Barros[†]

2021-09-13

Contents

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

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

[†]<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` 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, 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 / label }
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 / label }
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 / label }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 and 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 / reference }
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 and noabbrevfirst options

```

450 \bool_new:N \l__zrefclever_abbrev_bool
451 \bool_new:N \l__zrefclever_noabbrev_first_bool
452 \keys_define:nn { zref-clever / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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 / reference }
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

```

660 \keys_define:nn { }
661 {
662   zref-clever / zcsetup .inherit:n = zref-clever / label ,
663   zref-clever / zcsetup .inherit:n = zref-clever / reference ,
664 }

```

\zcsetup Provide \zcsetup.

```

665 \NewDocumentCommand \zcsetup { m }
666 { \keys_set:nn { zref-clever / zcsetup } {#1} }

```

(End definition for \zcsetup.)

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

```

667 \RequirePackage { l3keys2e }
668 \ProcessKeysOptions { zref-clever / 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
669 \tl_new:N \l__zrefclever_setup_type_tl
670 \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.

```

671 \NewDocumentCommand \zcRefTypeSetup { m m }
672 {
673   \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
674   { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
675   \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
676   \keys_set:nn { zref-clever / typesetup } {#2}
677 }

```

(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.

```

678 \clist_map_inline:nn
679 {
680     tpairsep ,
681     tlistsep ,
682     tlastsep ,
683     notesep ,
684 }
685 {
686     \keys_define:nn { zref-clever / typesetup }
687     {
688         #1 .code:n =
689         {
690             \msg_warning:nnn { zref-clever } { option-not-type-specific } {#1}
691         } ,
692     }
693 }
```

Possibly or necessarily type-specific options.

```

694 \clist_map_inline:nn
695 {
696     % Possibly type-specific options.
697     namefont ,
698     namesep ,
699     pairsep ,
700     listsep ,
701     lastsep ,
702     rangesep ,
703     reffont ,
704     refpre ,
705     refpos ,
706     reffont-in ,
707     refpre-in ,
708     refpos-in ,
709     % Necessarily type-specific options.
710     Name-sg ,
711     name-sg ,
712     Name-pl ,
713     name-pl ,
714     Name-sg-ab ,
715     name-sg-ab ,
716     Name-pl-ab ,
717     name-pl-ab ,
```

```

718 }
719 {
720   \keys_define:nn { zref-clever / typesetup }
721   {
722     #1 .default:V = \c_novalue_tl ,
723     #1 .code:n =
724     {
725       \tl_if_novalue:nTF {##1}
726       {
727         \prop_remove:cn
728         { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
729         {#1}
730       }
731       {
732         \prop_put:cnn
733         { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
734         {#1} {##1}
735       }
736     } ,
737   }
738 }

```

5.2 \zcDeclareTranslations

\zcDeclareTranslations Provide \zcDeclareTranslations.

```

739 \NewDocumentCommand \zcDeclareTranslations { m m }
740 {
741   \tl_set:Nn \l__zrefclever_setup_language_tl {#1}
742   \tl_clear:N \l__zrefclever_setup_type_tl
743   \keys_set:nn { zref-clever / translations } {#2}
744 }

(End definition for \zcDeclareTranslations.)

745 \keys_define:nn { zref-clever / translations }
746 {
747   type .code:n =
748   {
749     \tl_if_empty:nTF {#1}
750     { \tl_clear:N \l__zrefclever_setup_type_tl }
751     {
752       \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
753       { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
754       \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
755     }
756   } ,
757 }

```

Not type-specific options.

```

758 \clist_map_inline:nn
759 {
760   tpairsep ,
761   tlistsep ,
762   tlastsep ,
763   notesep ,

```

```

764 }
765 {
766   \keys_define:nn { zref-clever / translations }
767   {
768     #1 .value_required:n = true ,
769     #1 .code:n =
770     {
771       \tl_if_empty:NTF \l__zrefclever_setup_type_tl
772       {
773         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
774         { zrefclever-default- #1 } {##1}
775       }
776       {
777         \msg_warning:nnn { zref-clever }
778         { option-not-type-specific } {#1}
779       }
780     } ,
781   }
782 }

```

Possibly type-specific options.

```

783 \clist_map_inline:nn
784 {
785   namesep ,
786   pairsep ,
787   listsep ,
788   lastsep ,
789   rangesep ,
790   refpre ,
791   refpos ,
792   refpre-in ,
793   refpos-in ,
794 }
795 {
796   \keys_define:nn { zref-clever / translations }
797   {
798     #1 .value_required:n = true ,
799     #1 .code:n =
800     {
801       \tl_if_empty:NTF \l__zrefclever_setup_type_tl
802       {
803         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
804         { zrefclever-default- #1 } {##1}
805       }
806       {
807         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
808         { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
809       }
810     } ,
811   }
812 }

```

Necessarily type-specific options.

```

813 \clist_map_inline:nn
814 {

```

```

815     Name-sg ,
816     name-sg ,
817     Name-pl ,
818     name-pl ,
819     Name-sg-ab ,
820     name-sg-ab ,
821     Name-pl-ab ,
822     name-pl-ab ,
823 }
824 {
825     \keys_define:nn { zref-clever / translations }
826     {
827         #1 .value_required:n = true ,
828         #1 .code:n =
829         {
830             \tl_if_empty:NTF \l__zrefclever_setup_type_tl
831             {
832                 \msg_warning:nnn { zref-clever }
833                 { option-only-type-specific } {#1}
834             }
835             {
836                 \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
837                 { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
838             }
839         } ,
840     }
841 }

```

6 \zcref

```

\zcref          \zcref{*}[\<options>]{\<labels>}

842 \NewDocumentCommand \zcref { s O { } m }
843 { \zref@wrapper@babel \__zrefclever_zcref:nnn {#3} {#1} {#2} }

```

(End definition for \zcref.)

```

\l__zrefclever_zcref_labels_seq
\l__zrefclever_link_star_bool
844 \seq_new:N \l__zrefclever_zcref_labels_seq
845 \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>}

846 \cs_new_protected:Npn \__zrefclever_zcref:nnn #1#2#3
847 {
848     \group_begin:
849     \keys_set:nn { zref-clever / reference } {#3}
850     \seq_set_from_clist:Nn \l__zrefclever_zcref_labels_seq {#1}

```



```

851 \bool_set:Nn \l__zrefclever_link_star_bool {#2}
852 % Integration with 'zref-check'.
853 \bool_lazy_and:nnT
854   { \l__zrefclever_zrefcheck_available_bool }
855   { \l__zrefclever_zceref_with_check_bool }
856   { \zrefcheck_zceref_beg_label: }
857 \bool_lazy_or:nnT
858   { \l__zrefclever_typeset_sort_bool }
859   { \l__zrefclever_typeset_range_bool }
860   { \__zrefclever_sort_labels: }
861 \__zrefclever_typeset_refs:
862 % Typeset \texttt{note}.
863 \l__zrefclever_noteseq_tl
864 \l__zrefclever_zceref_note_tl
865 % Integration with 'zref-check'.
866 \bool_lazy_and:nnT
867   { \l__zrefclever_zrefcheck_available_bool }
868   { \l__zrefclever_zceref_with_check_bool }
869   {
870     \zrefcheck_zceref_end_label_maybe:
871     \zrefcheck_zceref_run_checks_on_labels:n
872     { \l__zrefclever_zceref_labels_seq }
873   }
874 \group_end:
875 }

```

(End definition for __zrefclever_zceref:nnnn.)

7 \zcpageref

```

\zcpageref \zcpageref*[\<options>]{\<labels>}
876 \NewDocumentCommand \zcpageref { s O { } m }
877 {
878   \IfBooleanTF {#1}
879     { \zcref*[#2, ref = page] {#3} }
880     { \zcref [#2, ref = page] {#3} }
881 }

```

(End definition for \zcpageref.)

8 Sorting

```

882 \int_new:N \l__zrefclever_sort_prior_a_int
883 \int_new:N \l__zrefclever_sort_prior_b_int

```

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_a_tl 884 \tl_new:N \l__zrefclever_label_a_tl
\l__zrefclever_label_b_tl 885 \tl_new:N \l__zrefclever_label_b_tl
\l__zrefclever_label_type_a_tl 886 \tl_new:N \l__zrefclever_label_type_a_tl
\l__zrefclever_label_type_b_tl 887 \tl_new:N \l__zrefclever_label_type_b_tl
\l__zrefclever_label_enclcnt_a_tl 888 \tl_new:N \l__zrefclever_label_enclcnt_a_tl
\l__zrefclever_label_enclval_a_tl
\l__zrefclever_label_enclval_b_tl

```

```

889 \tl_new:N \l__zrefclever_label_enclcnt_b_tl
890 \tl_new:N \l__zrefclever_label_enclval_a_tl
891 \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.

```

892 \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.

```

893 \cs_new_protected:Npn \__zrefclever_sort_labels:
894 {

```

Store label types sequence.

```

895   \seq_clear:N \l__zrefclever_label_types_seq
896   \bool_if:NF \l__zrefclever_page_ref_bool
897   {
898     \seq_map_function:NN
899     \l__zrefclever_zcref_labels_seq \__zrefclever_label_type_put_new_right:n
900   }

```

Sort.

```

901   \seq_sort:Nn \l__zrefclever_zcref_labels_seq
902   {
903     \zref@ifrefundefined {##1}
904     {
905       \zref@ifrefundefined {##2}
906       {
907         % Neither label is defined.
908         \sort_return_same:
909       }
910       {
911         % The second label is defined, but the first isn't, leave the
912         % undefined first (to be more visible).
913         \sort_return_same:
914       }
915     }
916     {
917       \zref@ifrefundefined {##2}
918       {
919         % The first label is defined, but the second isn't, bring the
920         % second forward.
921         \sort_return_swapped:
922       }
923       {
924         % The interesting case: both labels are defined. The
925         % reference to the "default" property/counter or to the page

```

```

926         % are quite different from our perspective, they rely on
927         % different fields and even use different information for
928         % sorting, so we branch them here to specialized functions.
929         \bool_if:NTF \l__zrefclever_page_ref_bool
930         { \__zrefclever_sort_page:nn {##1} {##2} }
931         { \__zrefclever_sort_default:nn {##1} {##2} }
932     }
933 }
934 }
935 }

```

(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>}

936 \cs_new_protected:Npn \__zrefclever_label_type_put_new_right:n #1
937 {
938     \tl_set:Nx \l__zrefclever_label_type_a_tl
939     { \zref@extractdefault {#1} { zc@type } { \c_empty_tl } }
940     \tl_if_empty:NF \l__zrefclever_label_type_a_tl
941     {
942         \seq_if_in:NVF \l__zrefclever_label_types_seq \l__zrefclever_label_type_a_tl
943         {
944             \seq_put_right:NV
945             \l__zrefclever_label_types_seq \l__zrefclever_label_type_a_tl
946         }
947     }
948 }

```

(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.

```

949 \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.

```

950 \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>}

```

```

951 \cs_new_protected:Npn \__zrefclever_sort_default:nn #1#2
952 {
953   \tl_set:Nx \l__zrefclever_label_type_a_tl
954     { \zref@extractdefault {#1} {zc@type} { \c_empty_tl } }
955   \tl_set:Nx \l__zrefclever_label_type_b_tl
956     { \zref@extractdefault {#2} {zc@type} { \c_empty_tl } }
957
958   \bool_if:nTF
959     {
960       % The second label has a type, but the first doesn't, leave the
961       % undefined first (to be more visible).
962       \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
963       ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
964     }
965     { \sort_return_same: }
966     {
967       \bool_if:nTF
968         {
969           % The first label has a type, but the second doesn't, bring the
970           % second forward.
971           ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
972           \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
973         }
974         { \sort_return_swapped: }
975         {
976           \bool_if:nTF
977             {
978               % The interesting case: both labels have a type\dots{}
979               ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
980               ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
981             }
982             {
983               % Here we send this to a couple of auxiliary functions for no
984               % other reason than to keep this long function a little less
985               % unreadable.
986               \tl_if_eq:NNTF \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
987                 {
988                   % \dots{} and it's the same type.
989                   \__zrefclever_sort_default_same_type:nn {#1} {#2}
990                 }
991                 {
992                   % \dots{} and they are different types.
993                   \__zrefclever_sort_default_different_types:nn {#1} {#2}
994                 }
995             }
996         }
997         {
998           % Neither of the labels has a type. We can't do much of
999           % meaningful here, but if it's the same counter, compare it.
1000           \exp_args:Nxx \tl_if_eq:nnTF
1001             { \zref@extractdefault {#1} { counter } { } }
1002             { \zref@extractdefault {#2} { counter } { } }
1003             {
1004               \int_compare:nNnTF

```

```

1005         >
1006         { \zref@extractdefault {#2} { zc@cntval } {-1} }
1007         { \sort_return_swapped: }
1008         { \sort_return_same: }
1009     }
1010     { \sort_return_same: }
1011 }
1012 }
1013 }
1014 }

```

(End definition for _zrefclever_sort_default:nn.)

_zrefclever_sort_default_same_type:nn

```

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

```

```

1055 { \zref@extractdefault {#2} { counter } { } }
1056 {
1057   \bool_set_true:N \l__zrefclever_sort_decided_bool
1058   \int_compare:nNnTF
1059     { \zref@extractdefault {#1} { zc@cntval } {-1} }
1060     >
1061     { \zref@extractdefault {#2} { zc@cntval } {-1} }
1062     { \sort_return_swapped: }
1063     { \sort_return_same: }
1064   }
1065   {
1066     \msg_warning:nnnn { zref-clever }
1067     { counters-not-nested } {#1} {#2}
1068     \bool_set_true:N \l__zrefclever_sort_decided_bool
1069     \sort_return_same:
1070   }
1071 }
1072 {
1073   \bool_if:nTF
1074   {
1075     % 'a' is empty (and 'b' is not), meaning: 'b' is (possibly)
1076     % nested in 'a'.
1077     \tl_if_empty_p:V \l_tmpa_tl
1078   }
1079   {
1080     \tl_set:Nx \l_tmpa_tl
1081     { {\zref@extractdefault {#1} { counter } { }} }
1082     \exp_args:NNx \tl_if_in:NnTF
1083     \l__zrefclever_label_enclcnt_b_tl { \l_tmpa_tl }
1084     {
1085       \bool_set_true:N \l__zrefclever_sort_decided_bool
1086       \sort_return_same:
1087     }
1088     {
1089       \msg_warning:nnnn { zref-clever }
1090       { counters-not-nested } {#1} {#2}
1091       \bool_set_true:N \l__zrefclever_sort_decided_bool
1092       \sort_return_same:
1093     }
1094   }
1095   {
1096     \bool_if:nTF
1097     {
1098       % 'b' is empty (and 'a' is not), meaning: 'a' is
1099       % (possibly) nested in 'b'.
1100       \tl_if_empty_p:V \l_tmpb_tl
1101     }
1102     {
1103       \tl_set:Nx \l_tmpb_tl
1104       { {\zref@extractdefault {#2} { counter } { }} }
1105       \exp_args:NNx \tl_if_in:NnTF
1106       \l__zrefclever_label_enclcnt_a_tl { \l_tmpb_tl }
1107       {
1108         \bool_set_true:N \l__zrefclever_sort_decided_bool

```

```

1109         \sort_return_swapped:
1110     }
1111     {
1112         \msg_warning:nnnn { zref-clever }
1113         { counters-not-nested } {#1} {#2}
1114         \bool_set_true:N \l__zrefclever_sort_decided_bool
1115         \sort_return_same:
1116     }
1117 }
1118 {
1119     % Neither is empty, meaning: we can (possibly) compare the
1120     % values of the current enclosing counter in the loop, if
1121     % they are equal, we are still in the loop, if they are
1122     % not, a sorting decision can be made directly.
1123     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
1124     {
1125         \int_compare:nNnTF
1126             { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1127             =
1128             { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1129             {
1130                 \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1131                     { \tl_tail:N \l__zrefclever_label_enclcnt_a_tl }
1132                 \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1133                     { \tl_tail:N \l__zrefclever_label_enclcnt_b_tl }
1134                 \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1135                     { \tl_tail:N \l__zrefclever_label_enclval_a_tl }
1136                 \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1137                     { \tl_tail:N \l__zrefclever_label_enclval_b_tl }
1138             }
1139             {
1140                 \bool_set_true:N \l__zrefclever_sort_decided_bool
1141                 \int_compare:nNnTF
1142                     { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1143                     >
1144                     { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1145                     { \sort_return_swapped: }
1146                     { \sort_return_same: }
1147             }
1148         }
1149     {
1150         \msg_warning:nnnn { zref-clever }
1151         { counters-not-nested } {#1} {#2}
1152         \bool_set_true:N \l__zrefclever_sort_decided_bool
1153         \sort_return_same:
1154     }
1155 }
1156 }
1157 }
1158 }
1159 }

```

(End definition for __zrefclever_sort_default_same_type:nn.)

_zrefclever_sort_default_different_types:nn

```

1160 \cs_new_protected:Npn \__zrefclever_sort_default_different_types:nn #1#2
1161 {
1162   \int_zero:N \l__zrefclever_sort_prior_a_int
1163   \int_zero:N \l__zrefclever_sort_prior_b_int
1164   % \cs{l__zrefclever_typesort_seq} was stored in reverse sequence, and we compute
1165   % the sort priorities in the negative range, so that we can implicitly
1166   % rely on '0' being the 'last value'.
1167   \seq_map_indexed_inline:Nn \l__zrefclever_typesort_seq
1168   {
1169     \tl_if_eq:nnTF {##2} {{othertypes}}
1170     {
1171       \int_compare:nNnT { \l__zrefclever_sort_prior_a_int } = { 0 }
1172       { \int_set:Nn \l__zrefclever_sort_prior_a_int { - ##1 } }
1173       \int_compare:nNnT { \l__zrefclever_sort_prior_b_int } = { 0 }
1174       { \int_set:Nn \l__zrefclever_sort_prior_b_int { - ##1 } }
1175     }
1176     {
1177       \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##2}
1178       { \int_set:Nn \l__zrefclever_sort_prior_a_int { - ##1 } }
1179       {
1180         \tl_if_eq:NnT \l__zrefclever_label_type_b_tl {##2}
1181         { \int_set:Nn \l__zrefclever_sort_prior_b_int { - ##1 } }
1182       }
1183     }
1184   }
1185   \bool_if:nTF
1186   {
1187     \int_compare_p:nNn
1188     { \l__zrefclever_sort_prior_a_int } <
1189     { \l__zrefclever_sort_prior_b_int }
1190   }
1191   { \sort_return_same: }
1192   {
1193     \bool_if:nTF
1194     {
1195       \int_compare_p:nNn
1196       { \l__zrefclever_sort_prior_a_int } >
1197       { \l__zrefclever_sort_prior_b_int }
1198     }
1199     { \sort_return_swapped: }
1200     {
1201       % Sort priorities are equal for different types: the type that
1202       % occurs first in \meta{labels}, as given by the user, is kept (or
1203       % brought) forward.
1204       \seq_map_inline:Nn \l__zrefclever_label_types_seq
1205       {
1206         \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##1}
1207         { \seq_map_break:n { \sort_return_same: } }
1208         {
1209           \tl_if_eq:NnT \l__zrefclever_label_type_b_tl {##1}
1210           { \seq_map_break:n { \sort_return_swapped: } }
1211         }
1212       }
1213     }
1214   }

```



```

1213     }
1214   }
1215 }

```

(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}

1216 \cs_new_protected:Npn \__zrefclever_sort_page:nn #1#2
1217 {
1218   \int_compare:nNnTF
1219     { \zref@extractdefault {#1} { abspage } {-1} }
1220     >
1221     { \zref@extractdefault {#2} { abspage } {-1} }
1222     { \sort_return_swapped: }
1223     { \sort_return_same:   }
1224 }

```

(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` 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.

```

1225 \bool_new:N \l__zrefclever_typeset_last_bool
1226 \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.

```

1227 \seq_new:N \l__zrefclever_typeset_labels_seq
1228 \tl_new:N \l__zrefclever_typeset_queue_prev_tl
1229 \tl_new:N \l__zrefclever_typeset_queue_curr_tl
1230 \tl_new:N \l__zrefclever_type_first_label_tl
1231 \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
\l_zrefclever_type_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.

```

1232 \int_new:N \l__zrefclever_label_count_int
1233 \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
\l_zrefclever_range_same_count_int
\l_zrefclever_range_beg_label_tl
\l_zrefclever_next_maybe_range_bool
\l_zrefclever_next_is_same_bool
\l_zrefclever_range_inhibit_next_bool

```

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.

```

1234 \int_new:N \l__zrefclever_range_count_int
1235 \int_new:N \l__zrefclever_range_same_count_int
1236 \tl_new:N \l__zrefclever_range_beg_label_tl
1237 \bool_new:N \l__zrefclever_next_maybe_range_bool
1238 \bool_new:N \l__zrefclever_next_is_same_bool
1239 \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.

```

1240 \tl_new:N \l__zrefclever_namefont_tl
1241 \tl_new:N \l__zrefclever_reffont_out_tl
1242 \tl_new:N \l__zrefclever_reffont_in_tl
1243
1244 \tl_new:N \l__zrefclever_namesep_tl
1245 \tl_new:N \l__zrefclever_rangesep_tl
1246 \tl_new:N \l__zrefclever_pairsep_tl
1247 \tl_new:N \l__zrefclever_listsep_tl
1248 \tl_new:N \l__zrefclever_lastsep_tl

```

```

1249 % 't' for 'type'
1250 \tl_new:N \l__zrefclever_tpairsep_tl
1251 \tl_new:N \l__zrefclever_tlistsep_tl
1252 \tl_new:N \l__zrefclever_tlastsep_tl
1253 \tl_new:N \l__zrefclever_notesep_tl
1254 \tl_new:N \l__zrefclever_refpre_out_tl
1255 \tl_new:N \l__zrefclever_refpos_out_tl
1256 \tl_new:N \l__zrefclever_refpre_in_tl
1257 \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_name_setup:.

```

1258 \tl_new:N \l__zrefclever_type_name_tl
1259 \bool_new:N \l__zrefclever_name_in_link_bool
1260 \tl_new:N \l__zrefclever_name_format_tl
1261 \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 \zceref.

```

1262 \cs_new_protected:Npn \__zrefclever_typeset_refs:
1263 {
1264   \seq_set_eq:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_zceref_labels_seq
1265   \tl_clear:N \l__zrefclever_typeset_queue_prev_tl
1266   \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1267   \tl_clear:N \l__zrefclever_type_first_label_tl
1268   \tl_clear:N \l__zrefclever_type_first_label_type_tl
1269   \tl_clear:N \l__zrefclever_range_beg_label_tl
1270   \int_zero:N \l__zrefclever_label_count_int
1271   \int_zero:N \l__zrefclever_type_count_int
1272   \int_zero:N \l__zrefclever_range_count_int
1273   \int_zero:N \l__zrefclever_range_same_count_int
1274
1275   % Get not-type-specific separators and refpre/pos options.
1276   \__zrefclever_get_option_with_transl:nN {tpairsep} \l__zrefclever_tpairsep_tl
1277   \__zrefclever_get_option_with_transl:nN {tlistsep} \l__zrefclever_tlistsep_tl
1278   \__zrefclever_get_option_with_transl:nN {tlastsep} \l__zrefclever_tlastsep_tl
1279   \__zrefclever_get_option_with_transl:nN {notesep} \l__zrefclever_notesep_tl
1280
1281   % Set the font option for this zceref call.
1282   \l__zrefclever_ref_typeset_font_tl
1283
1284   % Loop over the label list in sequence.
1285   \bool_set_false:N \l__zrefclever_typeset_last_bool
1286   \bool_until_do:Nn \l__zrefclever_typeset_last_bool
1287   {
1288     \seq_pop_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_a_tl
1289     \seq_if_empty:NTF \l__zrefclever_typeset_labels_seq
1290     {
1291       \tl_clear:N \l__zrefclever_label_b_tl

```

```

1292         \bool_set_true:N \l__zrefclever_typeset_last_bool
1293     }
1294     { \seq_get_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_b_tl }
1295
1296 \bool_if:NTF \l__zrefclever_page_ref_bool
1297 {
1298     \tl_set:Nn \l__zrefclever_label_type_a_tl { page }
1299     \tl_set:Nn \l__zrefclever_label_type_b_tl { page }
1300 }
1301 {
1302     \tl_set:Nx \l__zrefclever_label_type_a_tl
1303     {
1304         \zref@extractdefault
1305         { \l__zrefclever_label_a_tl } { zc@type } { \c_empty_tl }
1306     }
1307     \tl_set:Nx \l__zrefclever_label_type_b_tl
1308     {
1309         \zref@extractdefault
1310         { \l__zrefclever_label_b_tl } { zc@type } { \c_empty_tl }
1311     }
1312 }
1313
1314 % First, we establish whether the ‘‘current label’’ (i.e. ‘a’) is the
1315 % last one of its type. This can happen because the ‘‘next label’’
1316 % (i.e. ‘b’) is of a different type (or different definition status),
1317 % or because we are at the end of the list.
1318 \bool_if:NTF \l__zrefclever_typeset_last_bool
1319 { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1320 {
1321     \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1322     {
1323         \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1324         { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1325         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1326     }
1327     {
1328         \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1329         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1330         {
1331             % Neither is undefined, we must check the types.
1332             \bool_if:nTF
1333             % Both empty: same ‘‘type’’.
1334             {
1335                 \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1336                 \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1337             }
1338             { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1339             {
1340                 \bool_if:nTF
1341                 % Neither empty: compare types.
1342                 {
1343                     ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1344                     ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1345                 }

```

```

1346         {
1347             \tl_if_eq:NNTF
1348                 \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
1349                 { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1350                 { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1351         }
1352         % One empty, the other not: different ‘types’.
1353         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1354     }
1355 }
1356 }
1357 }
1358
1359 % Handle warnings in case of reference or type undefined.
1360 \zref@refused { \l__zrefclever_label_a_tl }
1361 \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1362 {}
1363 {
1364     \tl_if_empty:NT \l__zrefclever_label_type_a_tl
1365     {
1366         \msg_warning:nmx { zref-clever } { missing-type }
1367         { \l__zrefclever_label_a_tl }
1368     }
1369 }
1370
1371 % Get type-specific separators, refpre/pos and font options, once per
1372 % type.
1373 \int_compare:nNnT { \l__zrefclever_label_count_int } = { 0 }
1374 {
1375     \__zrefclever_get_option_plain:nN {namefont}          \l__zrefclever_namefont_tl
1376     \__zrefclever_get_option_plain:nN {reffont}          \l__zrefclever_reffont_out_tl
1377     \__zrefclever_get_option_plain:nN {reffont-in}       \l__zrefclever_reffont_in_tl
1378     \__zrefclever_get_option_with_transl:nN {namesep}    \l__zrefclever_namesep_tl
1379     \__zrefclever_get_option_with_transl:nN {rangesep}   \l__zrefclever_rangesep_tl
1380     \__zrefclever_get_option_with_transl:nN {pairsep}    \l__zrefclever_pairsep_tl
1381     \__zrefclever_get_option_with_transl:nN {listsep}    \l__zrefclever_listsep_tl
1382     \__zrefclever_get_option_with_transl:nN {lastsep}    \l__zrefclever_lastsep_tl
1383     \__zrefclever_get_option_with_transl:nN {refpre}     \l__zrefclever_refpre_out_tl
1384     \__zrefclever_get_option_with_transl:nN {refpos}     \l__zrefclever_refpos_out_tl
1385     \__zrefclever_get_option_with_transl:nN {refpre-in}  \l__zrefclever_refpre_in_tl
1386     \__zrefclever_get_option_with_transl:nN {refpos-in}  \l__zrefclever_refpos_in_tl
1387 }
1388
1389 % Here we send this to a couple of auxiliary functions for no other
1390 % reason than to keep this long function a little less unreadable.
1391 \bool_if:NTF \l__zrefclever_last_of_type_bool
1392 {
1393     % There exists no next label of the same type as the current.
1394     \__zrefclever_typeset_refs_aux_last_of_type:
1395 }
1396 {
1397     % There exists a next label of the same type as the current.
1398     \__zrefclever_typeset_refs_aux_not_last_of_type:
1399 }

```

```

1400     }
1401   }

```

(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.

```

1402 \cs_new_protected:Npn \_zrefclever_typeset_refs_aux_last_of_type:
1403 {
1404   % Process the current label to the current queue.
1405   \int_case:nnF { \l__zrefclever_label_count_int }
1406   {
1407     % It is the last label of its type, but also the first one, and that's
1408     % what matters here: just store it.
1409     { 0 }
1410     {
1411       \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1412       \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1413     }
1414
1415     % The last is the second: we have a pair (if not repeated).
1416     { 1 }
1417     {
1418       \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1419       {
1420         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1421         {
1422           \exp_not:V \l__zrefclever_pairsep_tl
1423           \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1424         }
1425       }
1426     }
1427   }
1428   % If neither the first, nor the second: we have the last label
1429   % on the current type list (if not repeated).
1430   {
1431     \int_case:nnF { \l__zrefclever_range_count_int }
1432     {
1433       % There was no range going on.
1434       {0}
1435       {
1436         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1437         {
1438           \exp_not:V \l__zrefclever_lastsep_tl
1439           \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1440         }
1441       }
1442       % Last in the range is also the second in it.
1443       {1}
1444       {
1445         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1446         {
1447           % We know 'range_beg_label' is not empty, since this is the
1448           % second element in the range, but the third or more in the
1449           % type list.

```

```

1450 \exp_not:V \l__zrefclever_listsep_tl
1451 \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1452 \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1453 {
1454   \exp_not:V \l__zrefclever_lastsep_tl
1455   \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1456 }
1457 }
1458 }
1459 }
1460 % Last in the range is third or more in it.
1461 {
1462   \int_case:nnF
1463   { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1464   {
1465     % Repetition, not a range.
1466     {0}
1467     {
1468       % If 'range_beg_label' is empty, it means it was also the
1469       % first of the type, and hence was already handled.
1470       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1471       {
1472         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1473         {
1474           \exp_not:V \l__zrefclever_lastsep_tl
1475           \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1476         }
1477       }
1478     }
1479     % A 'range', but with no skipped value, treat as list.
1480     {1}
1481     {
1482       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1483       {
1484         % Ditto.
1485         \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1486         {
1487           \exp_not:V \l__zrefclever_listsep_tl
1488           \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1489         }
1490         \exp_not:V \l__zrefclever_lastsep_tl
1491         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1492       }
1493     }
1494   }
1495   {
1496     % An actual range.
1497     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1498     {
1499       % Ditto.
1500       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1501       {
1502         \exp_not:V \l__zrefclever_lastsep_tl
1503         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl

```

```

1504         }
1505         \exp_not:V \l__zrefclever_rangesep_tl
1506         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1507     }
1508 }
1509 }
1510 }
1511
1512 % Handle ‘‘range’’ option. The idea is simple: if the queue is not empty,
1513 % we replace it with the end of the range (or pair). We can still
1514 % retrieve the end of the range from \cs{l__zrefclever_label_a_tl} since we know to
1515 % be processing the last label of its type at this point.
1516 \bool_if:NT \l__zrefclever_typeset_range_bool
1517 {
1518     \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1519     {
1520         \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1521         { }
1522         {
1523             \msg_warning:nxx { zref-clever } { single-element-range }
1524             { \l__zrefclever_type_first_label_type_tl }
1525         }
1526     }
1527     {
1528         \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1529         \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1530         { }
1531         {
1532             \__zrefclever_labels_in_sequence:nn
1533             { \l__zrefclever_type_first_label_tl } { \l__zrefclever_label_a_tl }
1534         }
1535         \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1536         {
1537             \bool_if:NTF \l__zrefclever_next_maybe_range_bool
1538             { \exp_not:V \l__zrefclever_pairsep_tl }
1539             { \exp_not:V \l__zrefclever_rangesep_tl }
1540             \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1541         }
1542     }
1543 }
1544
1545 % Now that the type is finished, we can add the name and the first ref to
1546 % the queue. Or, if ‘‘typset’’ option is not ‘‘both’’, handle it here
1547 % too.
1548 \__zrefclever_type_name_setup:
1549 \bool_if:nTF
1550 { \l__zrefclever_typeset_ref_bool && \l__zrefclever_typeset_name_bool }
1551 {
1552     \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1553     { \__zrefclever_get_ref_first: }
1554 }
1555 {
1556     \bool_if:nTF
1557     { \l__zrefclever_typeset_ref_bool }

```



```

1558 {
1559   \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1560   { \__zrefclever_get_ref:V \l__zrefclever_type_first_label_tl }
1561 }
1562 {
1563   \bool_if:nTF
1564   { \l__zrefclever_typeset_name_bool }
1565   {
1566     \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1567     {
1568       \bool_if:NTF \l__zrefclever_name_in_link_bool
1569       {
1570         \exp_not:N \group_begin:
1571         \exp_not:V \l__zrefclever_namefont_tl
1572         % It's two '@s', but escaped for DocStrip.
1573         \exp_not:N \hyper@@link
1574         {
1575           \zref@ifrefcontainsprop
1576           { \l__zrefclever_type_first_label_tl } { urluse }
1577           {
1578             \zref@extractdefault
1579             { \l__zrefclever_type_first_label_tl }
1580             { urluse } {}
1581           }
1582           {
1583             \zref@extractdefault
1584             { \l__zrefclever_type_first_label_tl }
1585             { url } {}
1586           }
1587         }
1588         {
1589           \zref@extractdefault
1590           { \l__zrefclever_type_first_label_tl } { anchor } {}
1591         }
1592         { \exp_not:V \l__zrefclever_type_name_tl }
1593         \exp_not:N \group_end:
1594       }
1595       {
1596         \exp_not:N \group_begin:
1597         \exp_not:V \l__zrefclever_namefont_tl
1598         \exp_not:V \l__zrefclever_type_name_tl
1599         \exp_not:N \group_end:
1600       }
1601     }
1602   }
1603   {
1604     % This case would correspond to "typeset=none" but should not
1605     % happen, given the options are set up to typeset at least one
1606     % of "ref" or "name", but a sensible fallback, equal to the
1607     % behavior of "both".
1608     \tl_put_left:Nx
1609     \l__zrefclever_typeset_queue_curr_tl { \__zrefclever_get_ref_first: }
1610   }
1611 }

```

```

1612     }
1613
1614 % Typeset the previous type, if there is one.
1615 \int_compare:nNnT { \l__zrefclever_type_count_int } > { 0 }
1616 {
1617     \int_compare:nNnT { \l__zrefclever_type_count_int } > { 1 }
1618     { \l__zrefclever_tlistsep_tl }
1619     \l__zrefclever_typeset_queue_prev_tl
1620 }
1621
1622 % Wrap up loop, or prepare for next iteration.
1623 \bool_if:NTF \l__zrefclever_typeset_last_bool
1624 {
1625     % We are finishing, typeset the current queue.
1626     \int_case:nnF { \l__zrefclever_type_count_int }
1627     {
1628         % Single type.
1629         { 0 }
1630         { \l__zrefclever_typeset_queue_curr_tl }
1631         % Pair of types.
1632         { 1 }
1633         {
1634             \l__zrefclever_tpairsep_tl
1635             \l__zrefclever_typeset_queue_curr_tl
1636         }
1637     }
1638     {
1639         % Last in list of types.
1640         \l__zrefclever_tlastsep_tl
1641         \l__zrefclever_typeset_queue_curr_tl
1642     }
1643 }
1644 {
1645     % There are further labels, set variables for next iteration.
1646     \tl_set_eq:NN
1647         \l__zrefclever_typeset_queue_prev_tl \l__zrefclever_typeset_queue_curr_tl
1648     \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1649     \tl_clear:N \l__zrefclever_type_first_label_tl
1650     \tl_clear:N \l__zrefclever_type_first_label_type_tl
1651     \tl_clear:N \l__zrefclever_range_beg_label_tl
1652     \int_zero:N \l__zrefclever_label_count_int
1653     \int_incr:N \l__zrefclever_type_count_int
1654     \int_zero:N \l__zrefclever_range_count_int
1655     \int_zero:N \l__zrefclever_range_same_count_int
1656 }
1657 }

```

(End definition for __zrefclever_typeset_refs_aux_last_of_type:.)

efclever_typeset_refs_aux_not_last_of_type: Handles typesetting of when the current label is not the last of its type.

```

1658 \cs_new_protected:Npn \__zrefclever_typeset_refs_aux_not_last_of_type:
1659 {
1660     % Signal if next label may form a range with the current one (of
1661     % course, only considered if compression is enabled in the first

```

```

1662 % place).
1663 \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1664 \bool_set_false:N \l__zrefclever_next_is_same_bool
1665 \bool_lazy_and:nnT
1666 { \l__zrefclever_typeset_compress_bool }
1667 % Currently no-op, but kept as ‘handle’ to inhibit compression of
1668 % individual labels.
1669 { ! \l__zrefclever_range_inhibit_next_bool }
1670 {
1671   \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1672   { }
1673   {
1674     \__zrefclever_labels_in_sequence:nn
1675     { \l__zrefclever_label_a_tl } { \l__zrefclever_label_b_tl }
1676   }
1677 }
1678
1679 % Process the current label to the current queue.
1680 \int_compare:nNnTF { \l__zrefclever_label_count_int } = { 0 }
1681 {
1682   % Current label is the first of its type (also not the last, but it
1683   % doesn’t matter here): just store the label.
1684   \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1685   \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1686
1687   % If the next label may be part of a range, we set ‘range_beg_label’
1688   % to ‘empty’ (we deal with it as the ‘first’, and must do it
1689   % there, to handle hyperlinking), but also step the range counters.
1690   \bool_if:NT \l__zrefclever_next_maybe_range_bool
1691   {
1692     \tl_clear:N \l__zrefclever_range_beg_label_tl
1693     \int_incr:N \l__zrefclever_range_count_int
1694     \bool_if:NT \l__zrefclever_next_is_same_bool
1695     { \int_incr:N \l__zrefclever_range_same_count_int }
1696   }
1697 }
1698 {
1699   % Current label is neither the first (nor the last) of its
1700   % type.
1701   \bool_if:NNTF \l__zrefclever_next_maybe_range_bool
1702   {
1703     % Starting, or continuing a range.
1704     \int_compare:nNnTF
1705     { \l__zrefclever_range_count_int } = {0}
1706     {
1707       % There was no range going, we are starting one.
1708       \tl_set:NV \l__zrefclever_range_beg_label_tl \l__zrefclever_label_a_tl
1709       \int_incr:N \l__zrefclever_range_count_int
1710       \bool_if:NT \l__zrefclever_next_is_same_bool
1711       { \int_incr:N \l__zrefclever_range_same_count_int }
1712     }
1713     {
1714       % Second or more in the range, but not the last.
1715       \int_incr:N \l__zrefclever_range_count_int

```

```

1716         \bool_if:NT \l__zrefclever_next_is_same_bool
1717         { \int_incr:N \l__zrefclever_range_same_count_int }
1718     }
1719 }
1720 {
1721     % Next element is not in sequence, meaning: there was no range, or
1722     % we are closing one.
1723     \int_case:nnF { \l__zrefclever_range_count_int }
1724     {
1725         % There was no range going on.
1726         {0}
1727         {
1728             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1729             {
1730                 \exp_not:V \l__zrefclever_listsep_tl
1731                 \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1732             }
1733         }
1734         % Last is second in the range: if 'range_same_count' is also
1735         % '1', it's a repetition (drop it), otherwise, it's a "pair
1736         % within a list", treat as list.
1737         {1}
1738         {
1739             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1740             {
1741                 \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1742                 {
1743                     \exp_not:V \l__zrefclever_listsep_tl
1744                     \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1745                 }
1746                 \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1747                 {
1748                     \exp_not:V \l__zrefclever_listsep_tl
1749                     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1750                 }
1751             }
1752         }
1753     }
1754 }
1755 {
1756     % Last is third or more in the range: if 'range_count' and
1757     % 'range_same_count' are the same, its a repetition (drop it),
1758     % if they differ by '1', its a list, if they differ by more,
1759     % it is a real range.
1760     \int_case:nnF
1761     { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1762     {
1763         {0}
1764         {
1765             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1766             {
1767                 \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1768                 {
1769                     \exp_not:V \l__zrefclever_listsep_tl
1770                     \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl

```

```

1770     }
1771   }
1772 }
1773 {1}
1774 {
1775   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1776   {
1777     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1778     {
1779       \exp_not:V \l__zrefclever_listsep_tl
1780       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1781     }
1782     \exp_not:V \l__zrefclever_listsep_tl
1783     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1784   }
1785 }
1786 }
1787 {
1788   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1789   {
1790     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1791     {
1792       \exp_not:V \l__zrefclever_listsep_tl
1793       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1794     }
1795     \exp_not:V \l__zrefclever_rangesep_tl
1796     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1797   }
1798 }
1799 }
1800 % Reset counters.
1801 \int_zero:N \l__zrefclever_range_count_int
1802 \int_zero:N \l__zrefclever_range_same_count_int
1803 }
1804 }
1805 % Step label counter for next iteration.
1806 \int_incr:N \l__zrefclever_label_count_int
1807 }

```

(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.

```

1808 \cs_new:Npn \__zrefclever_get_ref:n #1
1809 {
1810   \zref@ifrefcontainsprop {#1} { \l__zrefclever_ref_property_tl }
1811   {
1812     \bool_if:nTF

```

```

1813 { \l__zrefclever_use_hyperref_bool && ! \l__zrefclever_link_star_bool }
1814 {
1815   \exp_not:N \group_begin:
1816   \exp_not:V \l__zrefclever_reffont_out_tl
1817   \exp_not:V \l__zrefclever_refpre_out_tl
1818   \exp_not:N \group_begin:
1819   \exp_not:V \l__zrefclever_reffont_in_tl
1820   % It's two '@s', but escaped for DocStrip.
1821   \exp_not:N \hyper@@link
1822   {
1823     \zref@ifrefcontainsprop {#1} { urluse }
1824     { \zref@extractdefault {#1} { urluse } {} }
1825     { \zref@extractdefault {#1} { url } {} }
1826   }
1827   { \zref@extractdefault {#1} { anchor } {} }
1828   {
1829     \exp_not:V \l__zrefclever_refpre_in_tl
1830     \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1831     \exp_not:V \l__zrefclever_refpos_in_tl
1832   }
1833   \exp_not:N \group_end:
1834   \exp_not:V \l__zrefclever_refpos_out_tl
1835   \exp_not:N \group_end:
1836 }
1837 {
1838   \exp_not:N \group_begin:
1839   \exp_not:V \l__zrefclever_reffont_out_tl
1840   \exp_not:V \l__zrefclever_refpre_out_tl
1841   \exp_not:N \group_begin:
1842   \exp_not:V \l__zrefclever_reffont_in_tl
1843   \exp_not:V \l__zrefclever_refpre_in_tl
1844   \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1845   \exp_not:V \l__zrefclever_refpos_in_tl
1846   \exp_not:N \group_end:
1847   \exp_not:V \l__zrefclever_refpos_out_tl
1848   \exp_not:N \group_end:
1849 }
1850 }
1851 { \exp_not:N \zref@default }
1852 }
1853 \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.

```

1854 \cs_new_protected:Npn \__zrefclever_type_name_setup:
1855 {
1856   \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1857   { \tl_clear:N \l__zrefclever_type_name_tl }
1858   {
1859     \tl_if_empty:nTF \l__zrefclever_type_first_label_type_tl
1860     { \tl_clear:N \l__zrefclever_type_name_tl }
1861     {

```

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

```

1862 \bool_lazy_or:nnTF
1863 { \l__zrefclever_capitalize_bool }
1864 {
1865   \l__zrefclever_capitalize_first_bool &&
1866   \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1867 }
1868 { \tl_set:Nn \l__zrefclever_name_format_tl {Name} }
1869 { \tl_set:Nn \l__zrefclever_name_format_tl {name} }
1870 % If the queue is empty, we have a singular, otherwise, plural.
1871 \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1872 { \tl_put_right:Nn \l__zrefclever_name_format_tl { -sg } }
1873 { \tl_put_right:Nn \l__zrefclever_name_format_tl { -pl } }
1874 \bool_lazy_and:nnTF
1875 { \l__zrefclever_abbrev_bool }
1876 {
1877   ! \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 } ||
1878   ! \l__zrefclever_noabbrev_first_bool
1879 }
1880 {
1881   \tl_set:NV \l__zrefclever_name_format_fallback_tl \l__zrefclever_name_format
1882   \tl_put_right:Nn \l__zrefclever_name_format_tl { -ab }
1883 }
1884 { \tl_clear:N \l__zrefclever_name_format_fallback_tl }
1885
1886 \tl_if_empty:NTF \l__zrefclever_name_format_fallback_tl
1887 {
1888   \prop_get:cVNF
1889   { l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1890     \l__zrefclever_name_format_tl
1891     \l__zrefclever_type_name_tl
1892     {
1893       \__zrefclever_if_transl:xxTF
1894       { \l__zrefclever_ref_language_tl }
1895       {
1896         zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1897         \l__zrefclever_name_format_tl
1898       }
1899       {
1900         \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1901         { \l__zrefclever_ref_language_tl }
1902         {
1903           zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1904           \l__zrefclever_name_format_tl
1905         }
1906       }
1907     }
1908     {
1909       \tl_clear:N \l__zrefclever_type_name_tl
1910       \msg_warning:nxx { zref-clever } { missing-name }
1911       { \l__zrefclever_type_first_label_type_tl }
1912     }
1913   }
1914 {

```

```

1915 \prop_get:cVNF
1916 { l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1917 \l__zrefclever_name_format_tl
1918 \l__zrefclever_type_name_tl
1919 {
1920 \prop_get:cVNF
1921 { l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options
1922 \l__zrefclever_name_format_fallback_tl
1923 \l__zrefclever_type_name_tl
1924 {
1925 \__zrefclever_if_transl:xxTF
1926 { \l__zrefclever_ref_language_tl }
1927 {
1928 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1929 \l__zrefclever_name_format_tl
1930 }
1931 {
1932 \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1933 { \l__zrefclever_ref_language_tl }
1934 {
1935 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1936 \l__zrefclever_name_format_tl
1937 }
1938 }
1939 {
1940 \__zrefclever_if_transl:xxTF
1941 { \l__zrefclever_ref_language_tl }
1942 {
1943 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1944 \l__zrefclever_name_format_fallback_tl
1945 }
1946 {
1947 \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1948 { \l__zrefclever_ref_language_tl }
1949 {
1950 zrefclever-type- \l__zrefclever_type_first_label_type_tl
1951 \l__zrefclever_name_format_fallback_tl
1952 }
1953 }
1954 {
1955 \tl_clear:N \l__zrefclever_type_name_tl
1956 \msg_warning:nxx { zref-clever } { missing-name }
1957 { \l__zrefclever_type_first_label_type_tl }
1958 }
1959 }
1960 }
1961 }
1962 }
1963 }
1964 }
Signal whether the type name is to be included in the hyperlink or not.
1965 \bool_lazy_any:nTF
1966 {
1967 { ! \l__zrefclever_use_hyperref_bool }

```



```

1968     { \l__zrefclever_link_star_bool }
1969     { \tl_if_empty_p:N \l__zrefclever_type_name_tl }
1970     { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { false } }
1971   }
1972   { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1973   {
1974     \bool_lazy_any:nTF
1975     {
1976       { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { true } }
1977       {
1978         \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { tsingle } &&
1979         \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl
1980       }
1981       {
1982         \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { single } &&
1983         \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl &&
1984         \l__zrefclever_typeset_last_bool &&
1985         \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1986       }
1987     }
1988     { \bool_set_true:N \l__zrefclever_name_in_link_bool }
1989     { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1990   }
1991 }

```

(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.

```

1992 \cs_new:Npn \__zrefclever_get_ref_first:
1993 {
1994   \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1995   { \exp_not:N \zref@default }
1996   {
1997     \bool_if:NTF \l__zrefclever_name_in_link_bool
1998     {
1999       \zref@ifrefcontainsprop
2000       { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
2001       {
2002         % It's two '@s', but escaped for DocStrip.
2003         \exp_not:N \hyper@@link
2004         {
2005           \zref@ifrefcontainsprop
2006           { \l__zrefclever_type_first_label_tl } { urluse }
2007           {
2008             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2009             { urluse } {}
2010           }
2011         }
2012         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2013         { url } {}
2014       }
2015     }
2016   }

```

```

2016 {
2017   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2018   { anchor } {}
2019 }
2020 {
2021   \exp_not:N \group_begin:
2022   \exp_not:V \l__zrefclever_namefont_tl
2023   \exp_not:V \l__zrefclever_type_name_tl
2024   \exp_not:N \group_end:
2025   \exp_not:V \l__zrefclever_namesep_tl
2026   \exp_not:N \group_begin:
2027   \exp_not:V \l__zrefclever_reffont_out_tl
2028   \exp_not:V \l__zrefclever_refpre_out_tl
2029   \exp_not:N \group_begin:
2030   \exp_not:V \l__zrefclever_reffont_in_tl
2031   \exp_not:V \l__zrefclever_refpre_in_tl
2032   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2033   { \l__zrefclever_ref_property_tl } {}
2034   \exp_not:V \l__zrefclever_refpos_in_tl
2035   \exp_not:N \group_end:
2036   % hyperlink makes it's own group, we'd like to close the
2037   % 'refpre-out' group after 'refpos-out', but... we close
2038   % it here, and give the trailing 'refpos-out' its own
2039   % group. This will result that formatting given to
2040   % 'refpre-out' will not reach 'refpos-out', but I see no
2041   % alternative, and this has to be handled specially.
2042   \exp_not:N \group_end:
2043 }
2044 \exp_not:N \group_begin:
2045 % Ditto: special treatment.
2046 \exp_not:V \l__zrefclever_reffont_out_tl
2047 \exp_not:V \l__zrefclever_refpos_out_tl
2048 \exp_not:N \group_end:
2049 }
2050 {
2051   \exp_not:N \group_begin:
2052   \exp_not:V \l__zrefclever_namefont_tl
2053   \exp_not:V \l__zrefclever_type_name_tl
2054   \exp_not:N \group_end:
2055   \exp_not:V \l__zrefclever_namesep_tl
2056   \exp_not:N \zref@default
2057 }
2058 }
2059 {
2060   \tl_if_empty:NTF \l__zrefclever_type_name_tl
2061   {
2062     \exp_not:N \zref@default
2063     \exp_not:V \l__zrefclever_namesep_tl
2064   }
2065   {
2066     \exp_not:N \group_begin:
2067     \exp_not:V \l__zrefclever_namefont_tl
2068     \exp_not:V \l__zrefclever_type_name_tl
2069     \exp_not:N \group_end:

```

```

2070         \exp_not:V \l__zrefclever_namesep_tl
2071     }
2072 \zref@ifrefcontainsprop
2073 { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
2074 {
2075     \bool_if:nTF
2076     {
2077         \l__zrefclever_use_hyperref_bool &&
2078         ! \l__zrefclever_link_star_bool
2079     }
2080     {
2081         \exp_not:N \group_begin:
2082         \exp_not:V \l__zrefclever_reffont_out_tl
2083         \exp_not:V \l__zrefclever_refpre_out_tl
2084         \exp_not:N \group_begin:
2085         \exp_not:V \l__zrefclever_reffont_in_tl
2086         % It's two '@s', but escaped for DocStrip.
2087         \exp_not:N \hyper@@link
2088         {
2089             \zref@ifrefcontainsprop
2090             { \l__zrefclever_type_first_label_tl } { urluse }
2091             {
2092                 \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2093                 { urluse } {}
2094             }
2095             {
2096                 \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2097                 { url } {}
2098             }
2099         }
2100         {
2101             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2102             { anchor } {}
2103         }
2104         {
2105             \exp_not:V \l__zrefclever_refpre_in_tl
2106             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2107             { \l__zrefclever_ref_property_tl } {}
2108             \exp_not:V \l__zrefclever_refpos_in_tl
2109         }
2110         \exp_not:N \group_end:
2111         \exp_not:V \l__zrefclever_refpos_out_tl
2112         \exp_not:N \group_end:
2113     }
2114     {
2115         \exp_not:N \group_begin:
2116         \exp_not:V \l__zrefclever_reffont_out_tl
2117         \exp_not:V \l__zrefclever_refpre_out_tl
2118         \exp_not:N \group_begin:
2119         \exp_not:V \l__zrefclever_reffont_in_tl
2120         \exp_not:V \l__zrefclever_refpre_in_tl
2121         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2122         { \l__zrefclever_ref_property_tl } {}
2123         \exp_not:V \l__zrefclever_refpos_in_tl

```

```

2124         \exp_not:N \group_end:
2125         \exp_not:V \l__zrefclever_refpos_out_tl
2126         \exp_not:N \group_end:
2127     }
2128 }
2129 { \exp_not:N \zref@default }
2130 }
2131 }
2132 }

```

(End definition for __zrefclever_get_ref_first:.)

__zrefclever_get_option_with_transl:nN

```

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

```

```

2174         { zrefclever-default- #1 }
2175     }
2176 }
2177 }
2178 }

```

(End definition for _zrefclever_get_option_with_transl:nN.)

_zrefclever_get_option_plain:nN

```

2179 \cs_new_protected:Npn \_zrefclever_get_option_plain:nN #1#2
2180 {
2181     % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.
2182     \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2183     {
2184         % If not found, try the type specific options.
2185         \bool_lazy_and:nnTF
2186         { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2187         {
2188             \prop_if_exist_p:c
2189             { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2190         }
2191         {
2192             \prop_get:cnNF
2193             { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2194             { \tl_clear:N #2 }
2195         }
2196         { \tl_clear:N #2 }
2197     }
2198 }

```

(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”.

```

2199 \cs_new_protected:Npn \_zrefclever_labels_in_sequence:nn #1#2
2200 {
2201     \bool_if:NTF \l__zrefclever_page_ref_bool
2202     {
2203         \exp_args:Nxx \tl_if_eq:nnT
2204         { \zref@extractdefault {#1} { zc@pgfmt } { } }
2205         { \zref@extractdefault {#2} { zc@pgfmt } { } }
2206         {
2207             \int_compare:nNnTF
2208             { \zref@extractdefault {#1} { zc@pgval } {-2} + 1 }
2209             =
2210             { \zref@extractdefault {#2} { zc@pgval } {-1} }
2211             { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2212             {
2213                 \int_compare:nNnT
2214                 { \zref@extractdefault {#1} { zc@pgval } {-1} }
2215                 =
2216                 { \zref@extractdefault {#2} { zc@pgval } {-1} }
2217                 {

```

```

2218         \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2219         \bool_set_true:N \l__zrefclever_next_is_same_bool
2220     }
2221 }
2222 }
2223 }
2224 {
2225     \exp_args:Nxx \tl_if_eq:nnT
2226     { \zref@extractdefault {#1} { counter } { } }
2227     { \zref@extractdefault {#2} { counter } { } }
2228     {
2229         \exp_args:Nxx \tl_if_eq:nnT
2230         { \zref@extractdefault {#1} { zc@enclval } { } }
2231         { \zref@extractdefault {#2} { zc@enclval } { } }
2232         {
2233             \int_compare:nNnTF
2234             { \zref@extractdefault {#1} { zc@cntval } {-2} + 1 }
2235             =
2236             { \zref@extractdefault {#2} { zc@cntval } {-1} }
2237             { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2238             {
2239                 \int_compare:nNnTF
2240                 { \zref@extractdefault {#1} { zc@cntval } {-1} }
2241                 =
2242                 { \zref@extractdefault {#2} { zc@cntval } {-1} }
2243                 {
2244                     \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2245                     \bool_set_true:N \l__zrefclever_next_is_same_bool
2246                 }
2247             }
2248         }
2249     }
2250 }
2251 }

```

(End definition for `__zrefclever_labels_in_sequence:nn`.)

10 Special handling

This section is meant to aggregate any “special handling” needed for L^AT_EX 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’.

```

2252 \__zrefclever_declare_default_transl:nnn { fallback } { namesep } {\nobreakspace}
2253 \__zrefclever_declare_default_transl:nnn { fallback } { pairsep } { {,~}
2254 \__zrefclever_declare_default_transl:nnn { fallback } { listsep } { {,~}
2255 \__zrefclever_declare_default_transl:nnn { fallback } { lastsep } { {,~}
2256 \__zrefclever_declare_default_transl:nnn { fallback } { tpairsep } { {,~}
2257 \__zrefclever_declare_default_transl:nnn { fallback } { tlistsep } { {,~}
2258 \__zrefclever_declare_default_transl:nnn { fallback } { tlastsep } { {,~}
2259 \__zrefclever_declare_default_transl:nnn { fallback } { notesep } { {~}
2260 \__zrefclever_declare_default_transl:nnn { fallback } { rangesep } {\textendash}
2261 \__zrefclever_declare_default_transl:nnn { fallback } { refpre } {}
2262 \__zrefclever_declare_default_transl:nnn { fallback } { refpos } {}
2263 \__zrefclever_declare_default_transl:nnn { fallback } { refpre-in } {}
2264 \__zrefclever_declare_default_transl:nnn { fallback } { refpos-in } {}
2265 \endpackage
2266 \iflang-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.

```

2267 \ProvideDictionaryFor{English}{zref-clever}
2268
2269 \zcdicDefaultTransl{namesep}{\nobreakspace}
2270 \zcdicDefaultTransl{pairsep}{~and\nobreakspace}
2271 \zcdicDefaultTransl{listsep}{,~}
2272 \zcdicDefaultTransl{lastsep}{~and\nobreakspace}
2273 \zcdicDefaultTransl{tpairsep}{~and\nobreakspace}
2274 \zcdicDefaultTransl{tlistsep}{,~}
2275 \zcdicDefaultTransl{tlastsep}{,~and\nobreakspace}

```

2276 \zcDicDefaultTransl{notesep}{~}
 2277 \zcDicDefaultTransl{rangesep}{~to\nobreakspace}
 2278 \zcDicDefaultTransl{refpre}{}
 2279 \zcDicDefaultTransl{refpos}{}
 2280 \zcDicDefaultTransl{refpre-in}{}
 2281 \zcDicDefaultTransl{refpos-in}{}
 2282
 2283 \zcDicTypeTransl{part}{Name-sg}{Part}
 2284 \zcDicTypeTransl{part}{name-sg}{part}
 2285 \zcDicTypeTransl{part}{Name-pl}{Parts}
 2286 \zcDicTypeTransl{part}{name-pl}{parts}
 2287
 2288 \zcDicTypeTransl{chapter}{Name-sg}{Chapter}
 2289 \zcDicTypeTransl{chapter}{name-sg}{chapter}
 2290 \zcDicTypeTransl{chapter}{Name-pl}{Chapters}
 2291 \zcDicTypeTransl{chapter}{name-pl}{chapters}
 2292
 2293 \zcDicTypeTransl{section}{Name-sg}{Section}
 2294 \zcDicTypeTransl{section}{name-sg}{section}
 2295 \zcDicTypeTransl{section}{Name-pl}{Sections}
 2296 \zcDicTypeTransl{section}{name-pl}{sections}
 2297
 2298 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraph}
 2299 \zcDicTypeTransl{paragraph}{name-sg}{paragraph}
 2300 \zcDicTypeTransl{paragraph}{Name-pl}{Paragraphs}
 2301 \zcDicTypeTransl{paragraph}{name-pl}{paragraphs}
 2302 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}
 2303 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}
 2304 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}
 2305 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}
 2306
 2307 \zcDicTypeTransl{appendix}{Name-sg}{Appendix}
 2308 \zcDicTypeTransl{appendix}{name-sg}{appendix}
 2309 \zcDicTypeTransl{appendix}{Name-pl}{Appendices}
 2310 \zcDicTypeTransl{appendix}{name-pl}{appendices}
 2311
 2312 \zcDicTypeTransl{page}{Name-sg}{Page}
 2313 \zcDicTypeTransl{page}{name-sg}{page}
 2314 \zcDicTypeTransl{page}{Name-pl}{Pages}
 2315 \zcDicTypeTransl{page}{name-pl}{pages}
 2316 \zcDicTypeTransl{page}{name-sg-ab}{p.}
 2317 \zcDicTypeTransl{page}{name-pl-ab}{pp.}
 2318
 2319 \zcDicTypeTransl{line}{Name-sg}{Line}
 2320 \zcDicTypeTransl{line}{name-sg}{line}
 2321 \zcDicTypeTransl{line}{Name-pl}{Lines}
 2322 \zcDicTypeTransl{line}{name-pl}{lines}
 2323
 2324 \zcDicTypeTransl{figure}{Name-sg}{Figure}
 2325 \zcDicTypeTransl{figure}{name-sg}{figure}
 2326 \zcDicTypeTransl{figure}{Name-pl}{Figures}
 2327 \zcDicTypeTransl{figure}{name-pl}{figures}
 2328 \zcDicTypeTransl{figure}{Name-sg-ab}{Fig.}
 2329 \zcDicTypeTransl{figure}{name-sg-ab}{fig.}

2330 \zcDicTypeTransl{figure}{Name-pl-ab}{Figs.}
 2331 \zcDicTypeTransl{figure}{name-pl-ab}{figs.}
 2332
 2333 \zcDicTypeTransl{table}{Name-sg}{Table}
 2334 \zcDicTypeTransl{table}{name-sg}{table}
 2335 \zcDicTypeTransl{table}{Name-pl}{Tables}
 2336 \zcDicTypeTransl{table}{name-pl}{tables}
 2337
 2338 \zcDicTypeTransl{item}{Name-sg}{Item}
 2339 \zcDicTypeTransl{item}{name-sg}{item}
 2340 \zcDicTypeTransl{item}{Name-pl}{Items}
 2341 \zcDicTypeTransl{item}{name-pl}{items}
 2342
 2343 \zcDicTypeTransl{footnote}{Name-sg}{Footnote}
 2344 \zcDicTypeTransl{footnote}{name-sg}{footnote}
 2345 \zcDicTypeTransl{footnote}{Name-pl}{Footnotes}
 2346 \zcDicTypeTransl{footnote}{name-pl}{footnotes}
 2347
 2348 \zcDicTypeTransl{note}{Name-sg}{Note}
 2349 \zcDicTypeTransl{note}{name-sg}{note}
 2350 \zcDicTypeTransl{note}{Name-pl}{Notes}
 2351 \zcDicTypeTransl{note}{name-pl}{notes}
 2352
 2353 \zcDicTypeTransl{equation}{Name-sg}{Equation}
 2354 \zcDicTypeTransl{equation}{name-sg}{equation}
 2355 \zcDicTypeTransl{equation}{Name-pl}{Equations}
 2356 \zcDicTypeTransl{equation}{name-pl}{equations}
 2357 \zcDicTypeTransl{equation}{Name-sg-ab}{Eq.}
 2358 \zcDicTypeTransl{equation}{name-sg-ab}{eq.}
 2359 \zcDicTypeTransl{equation}{Name-pl-ab}{Eqs.}
 2360 \zcDicTypeTransl{equation}{name-pl-ab}{eqs.}
 2361 \zcDicTypeTransl{equation}{refpre-in}{(}
 2362 \zcDicTypeTransl{equation}{refpos-in}{)}
 2363
 2364 \zcDicTypeTransl{theorem}{Name-sg}{Theorem}
 2365 \zcDicTypeTransl{theorem}{name-sg}{theorem}
 2366 \zcDicTypeTransl{theorem}{Name-pl}{Theorems}
 2367 \zcDicTypeTransl{theorem}{name-pl}{theorems}
 2368
 2369 \zcDicTypeTransl{lemma}{Name-sg}{Lemma}
 2370 \zcDicTypeTransl{lemma}{name-sg}{lemma}
 2371 \zcDicTypeTransl{lemma}{Name-pl}{Lemmas}
 2372 \zcDicTypeTransl{lemma}{name-pl}{lemmas}
 2373
 2374 \zcDicTypeTransl{corollary}{Name-sg}{Corollary}
 2375 \zcDicTypeTransl{corollary}{name-sg}{corollary}
 2376 \zcDicTypeTransl{corollary}{Name-pl}{Corollaries}
 2377 \zcDicTypeTransl{corollary}{name-pl}{corollaries}
 2378
 2379 \zcDicTypeTransl{proposition}{Name-sg}{Proposition}
 2380 \zcDicTypeTransl{proposition}{name-sg}{proposition}
 2381 \zcDicTypeTransl{proposition}{Name-pl}{Propositions}
 2382 \zcDicTypeTransl{proposition}{name-pl}{propositions}
 2383

```

2384 \zcDicTypeTransl{definition}{Name-sg}{Definition}
2385 \zcDicTypeTransl{definition}{name-sg}{definition}
2386 \zcDicTypeTransl{definition}{Name-pl}{Definitions}
2387 \zcDicTypeTransl{definition}{name-pl}{definitions}
2388
2389 \zcDicTypeTransl{proof}{Name-sg}{Proof}
2390 \zcDicTypeTransl{proof}{name-sg}{proof}
2391 \zcDicTypeTransl{proof}{Name-pl}{Proofs}
2392 \zcDicTypeTransl{proof}{name-pl}{proofs}
2393
2394 \zcDicTypeTransl{result}{Name-sg}{Result}
2395 \zcDicTypeTransl{result}{name-sg}{result}
2396 \zcDicTypeTransl{result}{Name-pl}{Results}
2397 \zcDicTypeTransl{result}{name-pl}{results}
2398
2399 \zcDicTypeTransl{example}{Name-sg}{Example}
2400 \zcDicTypeTransl{example}{name-sg}{example}
2401 \zcDicTypeTransl{example}{Name-pl}{Examples}
2402 \zcDicTypeTransl{example}{name-pl}{examples}
2403
2404 \zcDicTypeTransl{remark}{Name-sg}{Remark}
2405 \zcDicTypeTransl{remark}{name-sg}{remark}
2406 \zcDicTypeTransl{remark}{Name-pl}{Remarks}
2407 \zcDicTypeTransl{remark}{name-pl}{remarks}
2408
2409 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithm}
2410 \zcDicTypeTransl{algorithm}{name-sg}{algorithm}
2411 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithms}
2412 \zcDicTypeTransl{algorithm}{name-pl}{algorithms}
2413
2414 \zcDicTypeTransl{listing}{Name-sg}{Listing}
2415 \zcDicTypeTransl{listing}{name-sg}{listing}
2416 \zcDicTypeTransl{listing}{Name-pl}{Listings}
2417 \zcDicTypeTransl{listing}{name-pl}{listings}
2418
2419 \zcDicTypeTransl{exercise}{Name-sg}{Exercise}
2420 \zcDicTypeTransl{exercise}{name-sg}{exercise}
2421 \zcDicTypeTransl{exercise}{Name-pl}{Exercises}
2422 \zcDicTypeTransl{exercise}{name-pl}{exercises}
2423
2424 \zcDicTypeTransl{solution}{Name-sg}{Solution}
2425 \zcDicTypeTransl{solution}{name-sg}{solution}
2426 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
2427 \zcDicTypeTransl{solution}{name-pl}{solutions}
2428 </lang-english>
2429 <*lang-german>

```

German

```

2430 \ProvideDictionaryFor{German}{zref-clever}
2431
2432 \zcDicDefaultTransl{namesep}{\nobreakspace}
2433 \zcDicDefaultTransl{pairsep}{\~und\nobreakspace}

```

2434 \zcDicDefaultTransl{listsep}{~,~}
 2435 \zcDicDefaultTransl{lastsep}{~und\nobreakspace}
 2436 \zcDicDefaultTransl{tpairsep}{~und\nobreakspace}
 2437 \zcDicDefaultTransl{tlistsep}{~,~}
 2438 \zcDicDefaultTransl{tlastsep}{~und\nobreakspace}
 2439 \zcDicDefaultTransl{notesep}{~}
 2440 \zcDicDefaultTransl{rangesep}{~bis\nobreakspace}
 2441
 2442 \zcDicTypeTransl{part}{Name-sg}{Teil}
 2443 \zcDicTypeTransl{part}{name-sg}{Teil}
 2444 \zcDicTypeTransl{part}{Name-pl}{Teile}
 2445 \zcDicTypeTransl{part}{name-pl}{Teile}
 2446
 2447 \zcDicTypeTransl{chapter}{Name-sg}{Kapitel}
 2448 \zcDicTypeTransl{chapter}{name-sg}{Kapitel}
 2449 \zcDicTypeTransl{chapter}{Name-pl}{Kapitel}
 2450 \zcDicTypeTransl{chapter}{name-pl}{Kapitel}
 2451
 2452 \zcDicTypeTransl{section}{Name-sg}{Abschnitt}
 2453 \zcDicTypeTransl{section}{name-sg}{Abschnitt}
 2454 \zcDicTypeTransl{section}{Name-pl}{Abschnitte}
 2455 \zcDicTypeTransl{section}{name-pl}{Abschnitte}
 2456
 2457 \zcDicTypeTransl{paragraph}{Name-sg}{Absatz}
 2458 \zcDicTypeTransl{paragraph}{name-sg}{Absatz}
 2459 \zcDicTypeTransl{paragraph}{Name-pl}{Absätze}
 2460 \zcDicTypeTransl{paragraph}{name-pl}{Absätze}
 2461
 2462 \zcDicTypeTransl{appendix}{Name-sg}{Anhang}
 2463 \zcDicTypeTransl{appendix}{name-sg}{Anhang}
 2464 \zcDicTypeTransl{appendix}{Name-pl}{Anhänge}
 2465 \zcDicTypeTransl{appendix}{name-pl}{Anhänge}
 2466
 2467 \zcDicTypeTransl{page}{Name-sg}{Seite}
 2468 \zcDicTypeTransl{page}{name-sg}{Seite}
 2469 \zcDicTypeTransl{page}{Name-pl}{Seiten}
 2470 \zcDicTypeTransl{page}{name-pl}{Seiten}
 2471
 2472 \zcDicTypeTransl{line}{Name-sg}{Zeile}
 2473 \zcDicTypeTransl{line}{name-sg}{Zeile}
 2474 \zcDicTypeTransl{line}{Name-pl}{Zeilen}
 2475 \zcDicTypeTransl{line}{name-pl}{Zeilen}
 2476
 2477 \zcDicTypeTransl{figure}{Name-sg}{Abbildung}
 2478 \zcDicTypeTransl{figure}{name-sg}{Abbildung}
 2479 \zcDicTypeTransl{figure}{Name-pl}{Abbildungen}
 2480 \zcDicTypeTransl{figure}{name-pl}{Abbildungen}
 2481 \zcDicTypeTransl{figure}{Name-sg-ab}{Abb.}
 2482 \zcDicTypeTransl{figure}{name-sg-ab}{Abb.}
 2483 \zcDicTypeTransl{figure}{Name-pl-ab}{Abb.}
 2484 \zcDicTypeTransl{figure}{name-pl-ab}{Abb.}
 2485
 2486 \zcDicTypeTransl{table}{Name-sg}{Tabelle}
 2487 \zcDicTypeTransl{table}{name-sg}{Tabelle}

2488 \zcDicTypeTransl{table}{Name-pl}{Tabellen}
 2489 \zcDicTypeTransl{table}{name-pl}{Tabellen}
 2490
 2491 \zcDicTypeTransl{item}{Name-sg}{Punkt}
 2492 \zcDicTypeTransl{item}{name-sg}{Punkt}
 2493 \zcDicTypeTransl{item}{Name-pl}{Punkte}
 2494 \zcDicTypeTransl{item}{name-pl}{Punkte}
 2495
 2496 \zcDicTypeTransl{footnote}{Name-sg}{Fußnote}
 2497 \zcDicTypeTransl{footnote}{name-sg}{Fußnote}
 2498 \zcDicTypeTransl{footnote}{Name-pl}{Fußnoten}
 2499 \zcDicTypeTransl{footnote}{name-pl}{Fußnoten}
 2500
 2501 \zcDicTypeTransl{note}{Name-sg}{Anmerkung}
 2502 \zcDicTypeTransl{note}{name-sg}{Anmerkung}
 2503 \zcDicTypeTransl{note}{Name-pl}{Anmerkungen}
 2504 \zcDicTypeTransl{note}{name-pl}{Anmerkungen}
 2505
 2506 \zcDicTypeTransl{equation}{Name-sg}{Gleichung}
 2507 \zcDicTypeTransl{equation}{name-sg}{Gleichung}
 2508 \zcDicTypeTransl{equation}{Name-pl}{Gleichungen}
 2509 \zcDicTypeTransl{equation}{name-pl}{Gleichungen}
 2510 \zcDicTypeTransl{equation}{refpre-in}{(
 2511 \zcDicTypeTransl{equation}{refpos-in}{)})
 2512
 2513 \zcDicTypeTransl{theorem}{Name-sg}{Theorem}
 2514 \zcDicTypeTransl{theorem}{name-sg}{Theorem}
 2515 \zcDicTypeTransl{theorem}{Name-pl}{Theoreme}
 2516 \zcDicTypeTransl{theorem}{name-pl}{Theoreme}
 2517
 2518 \zcDicTypeTransl{lemma}{Name-sg}{Lemma}
 2519 \zcDicTypeTransl{lemma}{name-sg}{Lemma}
 2520 \zcDicTypeTransl{lemma}{Name-pl}{Lemmata}
 2521 \zcDicTypeTransl{lemma}{name-pl}{Lemmata}
 2522
 2523 \zcDicTypeTransl{corollary}{Name-sg}{Korollar}
 2524 \zcDicTypeTransl{corollary}{name-sg}{Korollar}
 2525 \zcDicTypeTransl{corollary}{Name-pl}{Korollare}
 2526 \zcDicTypeTransl{corollary}{name-pl}{Korollare}
 2527
 2528 \zcDicTypeTransl{proposition}{Name-sg}{Satz}
 2529 \zcDicTypeTransl{proposition}{name-sg}{Satz}
 2530 \zcDicTypeTransl{proposition}{Name-pl}{Sätze}
 2531 \zcDicTypeTransl{proposition}{name-pl}{Sätze}
 2532
 2533 \zcDicTypeTransl{definition}{Name-sg}{Definition}
 2534 \zcDicTypeTransl{definition}{name-sg}{Definition}
 2535 \zcDicTypeTransl{definition}{Name-pl}{Definitionen}
 2536 \zcDicTypeTransl{definition}{name-pl}{Definitionen}
 2537
 2538 \zcDicTypeTransl{proof}{Name-sg}{Beweis}
 2539 \zcDicTypeTransl{proof}{name-sg}{Beweis}
 2540 \zcDicTypeTransl{proof}{Name-pl}{Beweise}
 2541 \zcDicTypeTransl{proof}{name-pl}{Beweise}

```

2542
2543 \zcDicTypeTransl{result}{Name-sg}{Ergebnis}
2544 \zcDicTypeTransl{result}{name-sg}{Ergebnis}
2545 \zcDicTypeTransl{result}{Name-pl}{Ergebnisse}
2546 \zcDicTypeTransl{result}{name-pl}{Ergebnisse}
2547
2548 \zcDicTypeTransl{example}{Name-sg}{Beispiel}
2549 \zcDicTypeTransl{example}{name-sg}{Beispiel}
2550 \zcDicTypeTransl{example}{Name-pl}{Beispiele}
2551 \zcDicTypeTransl{example}{name-pl}{Beispiele}
2552
2553 \zcDicTypeTransl{remark}{Name-sg}{Bemerkung}
2554 \zcDicTypeTransl{remark}{name-sg}{Bemerkung}
2555 \zcDicTypeTransl{remark}{Name-pl}{Bemerkungen}
2556 \zcDicTypeTransl{remark}{name-pl}{Bemerkungen}
2557
2558 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithmus}
2559 \zcDicTypeTransl{algorithm}{name-sg}{Algorithmus}
2560 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithmen}
2561 \zcDicTypeTransl{algorithm}{name-pl}{Algorithmen}
2562
2563 \zcDicTypeTransl{listing}{Name-sg}{Listing} % CHECK
2564 \zcDicTypeTransl{listing}{name-sg}{Listing} % CHECK
2565 \zcDicTypeTransl{listing}{Name-pl}{Listings} % CHECK
2566 \zcDicTypeTransl{listing}{name-pl}{Listings} % CHECK
2567
2568 \zcDicTypeTransl{exercise}{Name-sg}{Übungsaufgabe}
2569 \zcDicTypeTransl{exercise}{name-sg}{Übungsaufgabe}
2570 \zcDicTypeTransl{exercise}{Name-pl}{Übungsaufgaben}
2571 \zcDicTypeTransl{exercise}{name-pl}{Übungsaufgaben}
2572
2573 \zcDicTypeTransl{solution}{Name-sg}{Lösung}
2574 \zcDicTypeTransl{solution}{name-sg}{Lösung}
2575 \zcDicTypeTransl{solution}{Name-pl}{Lösungen}
2576 \zcDicTypeTransl{solution}{name-pl}{Lösungen}
2577 </lang-german>
2578 <*lang-french>

```

French

```

2579 \ProvideDictionaryFor{French}{zref-clever}
2580
2581 \zcDicDefaultTransl{namesep}{\nobreakspace}
2582 \zcDicDefaultTransl{pairsep}{~et\nobreakspace}
2583 \zcDicDefaultTransl{listsep}{~,~}
2584 \zcDicDefaultTransl{lastsep}{~et\nobreakspace}
2585 \zcDicDefaultTransl{tpairsep}{~et\nobreakspace}
2586 \zcDicDefaultTransl{tlistsep}{~,~}
2587 \zcDicDefaultTransl{tlastsep}{~et\nobreakspace}
2588 \zcDicDefaultTransl{notesep}{~}
2589 \zcDicDefaultTransl{rangesep}{~à\nobreakspace}
2590
2591 \zcDicTypeTransl{part}{Name-sg}{Partie}
2592 \zcDicTypeTransl{part}{name-sg}{partie}

```

2593 \zcDicTypeTransl{part}{Name-pl}{Parties}
 2594 \zcDicTypeTransl{part}{name-pl}{parties}
 2595
 2596 \zcDicTypeTransl{chapter}{Name-sg}{Chapitre}
 2597 \zcDicTypeTransl{chapter}{name-sg}{chapitre}
 2598 \zcDicTypeTransl{chapter}{Name-pl}{Chapitres}
 2599 \zcDicTypeTransl{chapter}{name-pl}{chapitres}
 2600
 2601 \zcDicTypeTransl{section}{Name-sg}{Section}
 2602 \zcDicTypeTransl{section}{name-sg}{section}
 2603 \zcDicTypeTransl{section}{Name-pl}{Sections}
 2604 \zcDicTypeTransl{section}{name-pl}{sections}
 2605
 2606 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraphe}
 2607 \zcDicTypeTransl{paragraph}{name-sg}{paragraphe}
 2608 \zcDicTypeTransl{paragraph}{Name-pl}{Paragraphes}
 2609 \zcDicTypeTransl{paragraph}{name-pl}{paragraphes}
 2610
 2611 \zcDicTypeTransl{appendix}{Name-sg}{Annexe}
 2612 \zcDicTypeTransl{appendix}{name-sg}{annexe}
 2613 \zcDicTypeTransl{appendix}{Name-pl}{Annexes}
 2614 \zcDicTypeTransl{appendix}{name-pl}{annexes}
 2615
 2616 \zcDicTypeTransl{page}{Name-sg}{Page}
 2617 \zcDicTypeTransl{page}{name-sg}{page}
 2618 \zcDicTypeTransl{page}{Name-pl}{Pages}
 2619 \zcDicTypeTransl{page}{name-pl}{pages}
 2620
 2621 \zcDicTypeTransl{line}{Name-sg}{Ligne}
 2622 \zcDicTypeTransl{line}{name-sg}{ligne}
 2623 \zcDicTypeTransl{line}{Name-pl}{Lignes}
 2624 \zcDicTypeTransl{line}{name-pl}{lignes}
 2625
 2626 \zcDicTypeTransl{figure}{Name-sg}{Figure}
 2627 \zcDicTypeTransl{figure}{name-sg}{figure}
 2628 \zcDicTypeTransl{figure}{Name-pl}{Figures}
 2629 \zcDicTypeTransl{figure}{name-pl}{figures}
 2630
 2631 \zcDicTypeTransl{table}{Name-sg}{Table}
 2632 \zcDicTypeTransl{table}{name-sg}{table}
 2633 \zcDicTypeTransl{table}{Name-pl}{Tables}
 2634 \zcDicTypeTransl{table}{name-pl}{tables}
 2635
 2636 \zcDicTypeTransl{item}{Name-sg}{Point}
 2637 \zcDicTypeTransl{item}{name-sg}{point}
 2638 \zcDicTypeTransl{item}{Name-pl}{Points}
 2639 \zcDicTypeTransl{item}{name-pl}{points}
 2640
 2641 \zcDicTypeTransl{footnote}{Name-sg}{Note}
 2642 \zcDicTypeTransl{footnote}{name-sg}{note}
 2643 \zcDicTypeTransl{footnote}{Name-pl}{Notes}
 2644 \zcDicTypeTransl{footnote}{name-pl}{notes}
 2645
 2646 \zcDicTypeTransl{note}{Name-sg}{Note}

2647 \zcDicTypeTransl{note}{name-sg}{note}
 2648 \zcDicTypeTransl{note}{Name-pl}{Notes}
 2649 \zcDicTypeTransl{note}{name-pl}{notes}
 2650
 2651 \zcDicTypeTransl{equation}{Name-sg}{Équation}
 2652 \zcDicTypeTransl{equation}{name-sg}{équation}
 2653 \zcDicTypeTransl{equation}{Name-pl}{Équations}
 2654 \zcDicTypeTransl{equation}{name-pl}{équations}
 2655 \zcDicTypeTransl{equation}{refpre-in}{(}
 2656 \zcDicTypeTransl{equation}{refpos-in}{)}
 2657
 2658 \zcDicTypeTransl{theorem}{Name-sg}{Théorème}
 2659 \zcDicTypeTransl{theorem}{name-sg}{théorème}
 2660 \zcDicTypeTransl{theorem}{Name-pl}{Théorèmes}
 2661 \zcDicTypeTransl{theorem}{name-pl}{théorèmes}
 2662
 2663 \zcDicTypeTransl{lemma}{Name-sg}{Lemme}
 2664 \zcDicTypeTransl{lemma}{name-sg}{lemme}
 2665 \zcDicTypeTransl{lemma}{Name-pl}{Lemmes}
 2666 \zcDicTypeTransl{lemma}{name-pl}{lemmes}
 2667
 2668 \zcDicTypeTransl{corollary}{Name-sg}{Corollaire}
 2669 \zcDicTypeTransl{corollary}{name-sg}{corollaire}
 2670 \zcDicTypeTransl{corollary}{Name-pl}{Corollaires}
 2671 \zcDicTypeTransl{corollary}{name-pl}{corollaires}
 2672
 2673 \zcDicTypeTransl{proposition}{Name-sg}{Proposition}
 2674 \zcDicTypeTransl{proposition}{name-sg}{proposition}
 2675 \zcDicTypeTransl{proposition}{Name-pl}{Propositions}
 2676 \zcDicTypeTransl{proposition}{name-pl}{propositions}
 2677
 2678 \zcDicTypeTransl{definition}{Name-sg}{Définition}
 2679 \zcDicTypeTransl{definition}{name-sg}{définition}
 2680 \zcDicTypeTransl{definition}{Name-pl}{Définitions}
 2681 \zcDicTypeTransl{definition}{name-pl}{définitions}
 2682
 2683 \zcDicTypeTransl{proof}{Name-sg}{Démonstration}
 2684 \zcDicTypeTransl{proof}{name-sg}{démonstration}
 2685 \zcDicTypeTransl{proof}{Name-pl}{Démonstrations}
 2686 \zcDicTypeTransl{proof}{name-pl}{démonstrations}
 2687
 2688 \zcDicTypeTransl{result}{Name-sg}{Résultat}
 2689 \zcDicTypeTransl{result}{name-sg}{résultat}
 2690 \zcDicTypeTransl{result}{Name-pl}{Résultats}
 2691 \zcDicTypeTransl{result}{name-pl}{résultats}
 2692
 2693 \zcDicTypeTransl{example}{Name-sg}{Exemple}
 2694 \zcDicTypeTransl{example}{name-sg}{exemple}
 2695 \zcDicTypeTransl{example}{Name-pl}{Exemples}
 2696 \zcDicTypeTransl{example}{name-pl}{exemples}
 2697
 2698 \zcDicTypeTransl{remark}{Name-sg}{Remarque}
 2699 \zcDicTypeTransl{remark}{name-sg}{remarque}
 2700 \zcDicTypeTransl{remark}{Name-pl}{Remarques}

2701 \zcDicTypeTransl{remark}{name-pl}{remarques}
 2702
 2703 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithme}
 2704 \zcDicTypeTransl{algorithm}{name-sg}{algorithme}
 2705 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithmes}
 2706 \zcDicTypeTransl{algorithm}{name-pl}{algorithmes}
 2707
 2708 \zcDicTypeTransl{listing}{Name-sg}{Liste}
 2709 \zcDicTypeTransl{listing}{name-sg}{liste}
 2710 \zcDicTypeTransl{listing}{Name-pl}{Listes}
 2711 \zcDicTypeTransl{listing}{name-pl}{listes}
 2712
 2713 \zcDicTypeTransl{exercise}{Name-sg}{Exercice}
 2714 \zcDicTypeTransl{exercise}{name-sg}{exercice}
 2715 \zcDicTypeTransl{exercise}{Name-pl}{Exercices}
 2716 \zcDicTypeTransl{exercise}{name-pl}{exercices}
 2717
 2718 \zcDicTypeTransl{solution}{Name-sg}{Solution}
 2719 \zcDicTypeTransl{solution}{name-sg}{solution}
 2720 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
 2721 \zcDicTypeTransl{solution}{name-pl}{solutions}
 2722 \langle /lang-french \rangle
 2723 \langle *lang-portuguese \rangle

Portuguese

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

2752 \zcDicTypeTransl{paragraph}{name-sg}{parágrafo}
 2753 \zcDicTypeTransl{paragraph}{Name-pl}{Parágrafos}
 2754 \zcDicTypeTransl{paragraph}{name-pl}{parágrafos}
 2755 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}
 2756 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}
 2757 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}
 2758 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}
 2759
 2760 \zcDicTypeTransl{appendix}{Name-sg}{Apêndice}
 2761 \zcDicTypeTransl{appendix}{name-sg}{apêndice}
 2762 \zcDicTypeTransl{appendix}{Name-pl}{Apêndices}
 2763 \zcDicTypeTransl{appendix}{name-pl}{apêndices}
 2764
 2765 \zcDicTypeTransl{page}{Name-sg}{Página}
 2766 \zcDicTypeTransl{page}{name-sg}{página}
 2767 \zcDicTypeTransl{page}{Name-pl}{Páginas}
 2768 \zcDicTypeTransl{page}{name-pl}{páginas}
 2769 \zcDicTypeTransl{page}{name-sg-ab}{p.}
 2770 \zcDicTypeTransl{page}{name-pl-ab}{pp.}
 2771
 2772 \zcDicTypeTransl{line}{Name-sg}{Linha}
 2773 \zcDicTypeTransl{line}{name-sg}{linha}
 2774 \zcDicTypeTransl{line}{Name-pl}{Linhas}
 2775 \zcDicTypeTransl{line}{name-pl}{linhas}
 2776
 2777 \zcDicTypeTransl{figure}{Name-sg}{Figura}
 2778 \zcDicTypeTransl{figure}{name-sg}{figura}
 2779 \zcDicTypeTransl{figure}{Name-pl}{Figuras}
 2780 \zcDicTypeTransl{figure}{name-pl}{figuras}
 2781 \zcDicTypeTransl{figure}{Name-sg-ab}{Fig.}
 2782 \zcDicTypeTransl{figure}{name-sg-ab}{fig.}
 2783 \zcDicTypeTransl{figure}{Name-pl-ab}{Figs.}
 2784 \zcDicTypeTransl{figure}{name-pl-ab}{figs.}
 2785
 2786 \zcDicTypeTransl{table}{Name-sg}{Tabela}
 2787 \zcDicTypeTransl{table}{name-sg}{tabela}
 2788 \zcDicTypeTransl{table}{Name-pl}{Tabelas}
 2789 \zcDicTypeTransl{table}{name-pl}{tabelas}
 2790
 2791 \zcDicTypeTransl{item}{Name-sg}{Item}
 2792 \zcDicTypeTransl{item}{name-sg}{item}
 2793 \zcDicTypeTransl{item}{Name-pl}{Itens}
 2794 \zcDicTypeTransl{item}{name-pl}{itens}
 2795
 2796 \zcDicTypeTransl{footnote}{Name-sg}{Nota}
 2797 \zcDicTypeTransl{footnote}{name-sg}{nota}
 2798 \zcDicTypeTransl{footnote}{Name-pl}{Notas}
 2799 \zcDicTypeTransl{footnote}{name-pl}{notas}
 2800
 2801 \zcDicTypeTransl{note}{Name-sg}{Nota}
 2802 \zcDicTypeTransl{note}{name-sg}{nota}
 2803 \zcDicTypeTransl{note}{Name-pl}{Notas}
 2804 \zcDicTypeTransl{note}{name-pl}{notas}
 2805

2806 \zcDicTypeTransl{equation}{Name-sg}{Equação}
 2807 \zcDicTypeTransl{equation}{name-sg}{equação}
 2808 \zcDicTypeTransl{equation}{Name-pl}{Equações}
 2809 \zcDicTypeTransl{equation}{name-pl}{equações}
 2810 \zcDicTypeTransl{equation}{Name-sg-ab}{Eq.}
 2811 \zcDicTypeTransl{equation}{name-sg-ab}{eq.}
 2812 \zcDicTypeTransl{equation}{Name-pl-ab}{Eqs.}
 2813 \zcDicTypeTransl{equation}{name-pl-ab}{eqs.}
 2814 \zcDicTypeTransl{equation}{refpre-in}{(
 2815 \zcDicTypeTransl{equation}{refpos-in}{)})
 2816
 2817 \zcDicTypeTransl{theorem}{Name-sg}{Teorema}
 2818 \zcDicTypeTransl{theorem}{name-sg}{teorema}
 2819 \zcDicTypeTransl{theorem}{Name-pl}{Teoremas}
 2820 \zcDicTypeTransl{theorem}{name-pl}{teoremas}
 2821
 2822 \zcDicTypeTransl{lemma}{Name-sg}{Lema}
 2823 \zcDicTypeTransl{lemma}{name-sg}{lema}
 2824 \zcDicTypeTransl{lemma}{Name-pl}{Lemas}
 2825 \zcDicTypeTransl{lemma}{name-pl}{lemas}
 2826
 2827 \zcDicTypeTransl{corollary}{Name-sg}{Corolário}
 2828 \zcDicTypeTransl{corollary}{name-sg}{corolário}
 2829 \zcDicTypeTransl{corollary}{Name-pl}{Corolários}
 2830 \zcDicTypeTransl{corollary}{name-pl}{corolários}
 2831
 2832 \zcDicTypeTransl{proposition}{Name-sg}{Proposição}
 2833 \zcDicTypeTransl{proposition}{name-sg}{proposição}
 2834 \zcDicTypeTransl{proposition}{Name-pl}{Proposições}
 2835 \zcDicTypeTransl{proposition}{name-pl}{proposições}
 2836
 2837 \zcDicTypeTransl{definition}{Name-sg}{Definição}
 2838 \zcDicTypeTransl{definition}{name-sg}{definição}
 2839 \zcDicTypeTransl{definition}{Name-pl}{Definições}
 2840 \zcDicTypeTransl{definition}{name-pl}{definições}
 2841
 2842 \zcDicTypeTransl{proof}{Name-sg}{Demonstração}
 2843 \zcDicTypeTransl{proof}{name-sg}{demonstração}
 2844 \zcDicTypeTransl{proof}{Name-pl}{Demonstrações}
 2845 \zcDicTypeTransl{proof}{name-pl}{demonstrações}
 2846
 2847 \zcDicTypeTransl{result}{Name-sg}{Resultado}
 2848 \zcDicTypeTransl{result}{name-sg}{resultado}
 2849 \zcDicTypeTransl{result}{Name-pl}{Resultados}
 2850 \zcDicTypeTransl{result}{name-pl}{resultados}
 2851
 2852 \zcDicTypeTransl{example}{Name-sg}{Exemplo}
 2853 \zcDicTypeTransl{example}{name-sg}{exemplo}
 2854 \zcDicTypeTransl{example}{Name-pl}{Exemplos}
 2855 \zcDicTypeTransl{example}{name-pl}{exemplos}
 2856
 2857 \zcDicTypeTransl{remark}{Name-sg}{Observação}
 2858 \zcDicTypeTransl{remark}{name-sg}{observação}
 2859 \zcDicTypeTransl{remark}{Name-pl}{Observações}

2860 \zcDicTypeTransl{remark}{name-pl}{observações}
 2861
 2862 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}
 2863 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}
 2864 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}
 2865 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}
 2866
 2867 \zcDicTypeTransl{listing}{Name-sg}{Listagem}
 2868 \zcDicTypeTransl{listing}{name-sg}{listagem}
 2869 \zcDicTypeTransl{listing}{Name-pl}{Listagens}
 2870 \zcDicTypeTransl{listing}{name-pl}{listagens}
 2871
 2872 \zcDicTypeTransl{exercise}{Name-sg}{Exercício}
 2873 \zcDicTypeTransl{exercise}{name-sg}{exercício}
 2874 \zcDicTypeTransl{exercise}{Name-pl}{Exercícios}
 2875 \zcDicTypeTransl{exercise}{name-pl}{exercícios}
 2876
 2877 \zcDicTypeTransl{solution}{Name-sg}{Solução}
 2878 \zcDicTypeTransl{solution}{name-sg}{solução}
 2879 \zcDicTypeTransl{solution}{Name-pl}{Soluções}
 2880 \zcDicTypeTransl{solution}{name-pl}{soluções}
 2881 </lang-portuguese>
 2882 <*lang-spanish>

Spanish

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

2911 \zcDicTypeTransl{paragraph}{name-sg}{párrafo}
 2912 \zcDicTypeTransl{paragraph}{Name-pl}{Párrafos}
 2913 \zcDicTypeTransl{paragraph}{name-pl}{párrafos}
 2914
 2915 \zcDicTypeTransl{appendix}{Name-sg}{Apéndice}
 2916 \zcDicTypeTransl{appendix}{name-sg}{apéndice}
 2917 \zcDicTypeTransl{appendix}{Name-pl}{Apéndices}
 2918 \zcDicTypeTransl{appendix}{name-pl}{apéndices}
 2919
 2920 \zcDicTypeTransl{page}{Name-sg}{Página}
 2921 \zcDicTypeTransl{page}{name-sg}{página}
 2922 \zcDicTypeTransl{page}{Name-pl}{Páginas}
 2923 \zcDicTypeTransl{page}{name-pl}{páginas}
 2924
 2925 \zcDicTypeTransl{line}{Name-sg}{Línea}
 2926 \zcDicTypeTransl{line}{name-sg}{línea}
 2927 \zcDicTypeTransl{line}{Name-pl}{Líneas}
 2928 \zcDicTypeTransl{line}{name-pl}{líneas}
 2929
 2930 \zcDicTypeTransl{figure}{Name-sg}{Figura}
 2931 \zcDicTypeTransl{figure}{name-sg}{figura}
 2932 \zcDicTypeTransl{figure}{Name-pl}{Figuras}
 2933 \zcDicTypeTransl{figure}{name-pl}{figuras}
 2934
 2935 \zcDicTypeTransl{table}{Name-sg}{Cuadro}
 2936 \zcDicTypeTransl{table}{name-sg}{cuadro}
 2937 \zcDicTypeTransl{table}{Name-pl}{Cuadros}
 2938 \zcDicTypeTransl{table}{name-pl}{cuadros}
 2939
 2940 \zcDicTypeTransl{item}{Name-sg}{Punto}
 2941 \zcDicTypeTransl{item}{name-sg}{punto}
 2942 \zcDicTypeTransl{item}{Name-pl}{Puntos}
 2943 \zcDicTypeTransl{item}{name-pl}{puntos}
 2944
 2945 \zcDicTypeTransl{footnote}{Name-sg}{Nota}
 2946 \zcDicTypeTransl{footnote}{name-sg}{nota}
 2947 \zcDicTypeTransl{footnote}{Name-pl}{Notas}
 2948 \zcDicTypeTransl{footnote}{name-pl}{notas}
 2949
 2950 \zcDicTypeTransl{note}{Name-sg}{Nota}
 2951 \zcDicTypeTransl{note}{name-sg}{nota}
 2952 \zcDicTypeTransl{note}{Name-pl}{Notas}
 2953 \zcDicTypeTransl{note}{name-pl}{notas}
 2954
 2955 \zcDicTypeTransl{equation}{Name-sg}{Ecuación}
 2956 \zcDicTypeTransl{equation}{name-sg}{ecuación}
 2957 \zcDicTypeTransl{equation}{Name-pl}{Ecuaciones}
 2958 \zcDicTypeTransl{equation}{name-pl}{ecuaciones}
 2959 \zcDicTypeTransl{equation}{refpre-in}{(}
 2960 \zcDicTypeTransl{equation}{refpos-in}{)}
 2961
 2962 \zcDicTypeTransl{theorem}{Name-sg}{Teorema}
 2963 \zcDicTypeTransl{theorem}{name-sg}{teorema}
 2964 \zcDicTypeTransl{theorem}{Name-pl}{Teoremas}

2965 \zcDicTypeTransl{theorem}{name-pl}{teoremas}
 2966
 2967 \zcDicTypeTransl{lemma}{Name-sg}{Lema}
 2968 \zcDicTypeTransl{lemma}{name-sg}{lema}
 2969 \zcDicTypeTransl{lemma}{Name-pl}{Lemas}
 2970 \zcDicTypeTransl{lemma}{name-pl}{lemas}
 2971
 2972 \zcDicTypeTransl{corollary}{Name-sg}{Corolario}
 2973 \zcDicTypeTransl{corollary}{name-sg}{corolario}
 2974 \zcDicTypeTransl{corollary}{Name-pl}{Corolarios}
 2975 \zcDicTypeTransl{corollary}{name-pl}{corolarios}
 2976
 2977 \zcDicTypeTransl{proposition}{Name-sg}{Proposición}
 2978 \zcDicTypeTransl{proposition}{name-sg}{proposición}
 2979 \zcDicTypeTransl{proposition}{Name-pl}{Proposiciones}
 2980 \zcDicTypeTransl{proposition}{name-pl}{proposiciones}
 2981
 2982 \zcDicTypeTransl{definition}{Name-sg}{Definición}
 2983 \zcDicTypeTransl{definition}{name-sg}{definición}
 2984 \zcDicTypeTransl{definition}{Name-pl}{Definiciones}
 2985 \zcDicTypeTransl{definition}{name-pl}{definiciones}
 2986
 2987 \zcDicTypeTransl{proof}{Name-sg}{Demostración}
 2988 \zcDicTypeTransl{proof}{name-sg}{demostración}
 2989 \zcDicTypeTransl{proof}{Name-pl}{Demostraciones}
 2990 \zcDicTypeTransl{proof}{name-pl}{demostraciones}
 2991
 2992 \zcDicTypeTransl{result}{Name-sg}{Resultado}
 2993 \zcDicTypeTransl{result}{name-sg}{resultado}
 2994 \zcDicTypeTransl{result}{Name-pl}{Resultados}
 2995 \zcDicTypeTransl{result}{name-pl}{resultados}
 2996
 2997 \zcDicTypeTransl{example}{Name-sg}{Ejemplo}
 2998 \zcDicTypeTransl{example}{name-sg}{ejemplo}
 2999 \zcDicTypeTransl{example}{Name-pl}{Ejemplos}
 3000 \zcDicTypeTransl{example}{name-pl}{ejemplos}
 3001
 3002 \zcDicTypeTransl{remark}{Name-sg}{Observación}
 3003 \zcDicTypeTransl{remark}{name-sg}{observación}
 3004 \zcDicTypeTransl{remark}{Name-pl}{Observaciones}
 3005 \zcDicTypeTransl{remark}{name-pl}{observaciones}
 3006
 3007 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}
 3008 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}
 3009 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}
 3010 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}
 3011
 3012 \zcDicTypeTransl{listing}{Name-sg}{Listado}
 3013 \zcDicTypeTransl{listing}{name-sg}{listado}
 3014 \zcDicTypeTransl{listing}{Name-pl}{Listados}
 3015 \zcDicTypeTransl{listing}{name-pl}{listados}
 3016
 3017 \zcDicTypeTransl{exercise}{Name-sg}{Ejercicio}
 3018 \zcDicTypeTransl{exercise}{name-sg}{ejercicio}

```

3019 \zcDicTypeTransl{exercise}{Name-pl}{Ejercicios}
3020 \zcDicTypeTransl{exercise}{name-pl}{ejercicios}
3021
3022 \zcDicTypeTransl{solution}{Name-sg}{Solución}
3023 \zcDicTypeTransl{solution}{name-sg}{solución}
3024 \zcDicTypeTransl{solution}{Name-pl}{Soluciones}
3025 \zcDicTypeTransl{solution}{name-pl}{soluciones}
3026 </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.

| | | |
|--|----------------|-------------------------------------|
| A | | 1319, 1325, 1329, 1350, 1353, 1988, |
| \AddToHook | | 2211, 2218, 2219, 2237, 2244, 2245 |
| . 91, 262, 282, 399, 472, 508, 534, 600 | | \bool_until_do:Nn 1038, 1286 |
| \appendix | 54 | |
| \appendixname | 55 | |
| \Arg | 2133 | |
| \AtEndOfPackage | 17, 506 | |
| B | | |
| \baselanguage | 497, 558, 574 | |
| bool commands: | | |
| \bool_case_true: | 2 | |
| \bool_if:NTF | | |
| .. 403, 407, 896, 929, 1296, 1318, | | |
| 1391, 1516, 1537, 1568, 1623, 1690, | | |
| 1694, 1701, 1710, 1716, 1997, 2201 | | |
| \bool_if:nTF ... 59, 958, 967, 976, | | |
| 1045, 1073, 1096, 1185, 1193, 1332, | | |
| 1340, 1549, 1556, 1563, 1812, 2075 | | |
| \bool_lazy_all:nTF | 2140 | |
| \bool_lazy_and:nnTF | | |
| 853, 866, 1665, 1874, 2185 | | |
| \bool_lazy_any:nTF | 1965, 1974 | |
| \bool_lazy_or:nnTF | 857, 1862 | |
| \bool_new:N | 246, | |
| 307, 308, 333, 360, 369, 376, 377, | | |
| 432, 433, 450, 451, 593, 594, 845, | | |
| 949, 1225, 1226, 1237, 1238, 1239, 1259 | | |
| \bool_set:Nn | 851 | |
| \bool_set_false:N .. 253, 267, 272, | | |
| 291, 302, 320, 324, 384, 393, 394, | | |
| 409, 615, 1034, 1285, 1324, 1338, | | |
| 1349, 1528, 1663, 1664, 1972, 1989 | | |
| \bool_set_true:N | | |
| . 258, 314, 315, 319, 325, 383, 388, | | |
| 389, 604, 609, 1057, 1068, 1085, | | |
| 1091, 1108, 1114, 1140, 1152, 1292, | | |
| C | | |
| clist commands: | | |
| \clist_map_inline:Nn | 544, 567 | |
| \clist_map_inline:nn | | |
| 200, 627, 678, 694, 758, 783, 813 | | |
| \counterwithin | 4 | |
| \cs 537, 1164, 1514, 2136, 2170, 2181 | | |
| cs commands: | | |
| \cs_generate_variant:Nn | | |
| 55, 56, 149, 152, 950, 1853 | | |
| \cs_if_exist:NTF | 39, 48, 69 | |
| \cs_new:Npn 37, 46, 57, 67, 78, 1808, 1992 | | |
| \cs_new_protected:Npn | | |
| 147, 150, 153, 161, 846, 893, | | |
| 936, 951, 1015, 1160, 1216, 1262, | | |
| 1402, 1658, 1854, 2134, 2179, 2199 | | |
| \cs_new_protected:Npx | 90 | |
| \cs_set_eq:NN | 94 | |
| D | | |
| \declaretranslation | 151, 154 | |
| \dots | 978, 988, 992 | |
| E | | |
| \endinput | 12 | |
| exp commands: | | |
| \exp_args:NNe | 27 | |
| \exp_args:NNx | 95, 1082, 1105 | |
| \exp_args:Nx 492, 496, 553, 557, 569, 573 | | |
| \exp_args:Nxx | | |
| 999, 1053, 2203, 2225, 2229 | | |
| \exp_not:N | 103, | |
| 109, 1570, 1573, 1593, 1596, 1599, | | |
| 1815, 1818, 1821, 1833, 1835, 1838, | | |

| | |
|--|--|
| 1841, 1846, 1848, 1851, 1995, 2003, 2021, 2024, 2026, 2029, 2035, 2042, 2044, 2048, 2051, 2054, 2056, 2062, 2066, 2069, 2081, 2084, 2087, 2110, 2112, 2115, 2118, 2124, 2126, 2129 | \int_set:Nn 1172, 1174, 1178, 1181 \int_use:N 33, 35, 50 \int_zero:N 1162, 1163, 1270, 1271, 1272, 1273, 1652, 1654, 1655, 1801, 1802 |
| \exp_not:n . 1422, 1438, 1450, 1454, 1474, 1487, 1490, 1502, 1505, 1538, 1539, 1571, 1592, 1597, 1598, 1730, 1743, 1748, 1768, 1779, 1782, 1792, 1795, 1816, 1817, 1819, 1829, 1831, 1834, 1839, 1840, 1842, 1843, 1845, 1847, 2022, 2023, 2025, 2027, 2028, 2030, 2031, 2034, 2046, 2047, 2052, 2053, 2055, 2063, 2067, 2068, 2070, 2082, 2083, 2085, 2105, 2108, 2111, 2116, 2117, 2119, 2120, 2123, 2125 | iow commands: \iow_newline: 120, 124 |
| | K |
| | keys commands: \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, 660, 686, 720, 745, 766, 796, 825 \keys_set:nn 21, 610, 666, 676, 743, 849 |
| | keyval commands: \keyval_parse:nnn 172, 226 |
| F | |
| file commands: \file_if_exist:nTF 492, 496, 553, 557, 569, 573 \fmtversion 3 | L \labelformat 3 \LoadDictionaryFor 494, 498, 555, 559, 571, 575, 581 |
| G | |
| group commands: \group_begin: 93, 848, 1570, 1596, 1815, 1818, 1838, 1841, 2021, 2026, 2029, 2044, 2051, 2066, 2081, 2084, 2115, 2118 \group_end: 96, 874, 1593, 1599, 1833, 1835, 1846, 1848, 2024, 2035, 2042, 2048, 2054, 2069, 2110, 2112, 2124, 2126 | M \MessageBreak 10 \meta 1202 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, 690, 777, 832, 1366, 1523, 1909, 1956 \msg_warning:nnnn 174, 1066, 1089, 1112, 1150 |
| H | |
| \hyperlink 45 | |
| I | |
| \IfBooleanTF 878 \IfFormatAtLeastTF 3, 4 \IfTranslation 142 | N \newcounter 4 \NewDocumentCommand 155, 157, 665, 671, 739, 842, 876 \NewHook 467 \newtheorem 55 \nobreakspace 2252, 2269, 2270, 2272, 2273, 2275, 2277, 2432, 2433, 2435, 2436, 2438, 2440, 2581, 2582, 2584, 2585, 2587, 2589, 2726, 2727, 2729, 2730, 2732, 2734, 2885, 2886, 2888, 2889, 2891, 2893 \noexpand 121 |
| int commands: \int_case:nnTF 1405, 1431, 1462, 1626, 1723, 1759 \int_compare:nNnTF 1003, 1058, 1125, 1141, 1171, 1173, 1218, 1373, 1418, 1452, 1615, 1617, 1680, 1704, 1746, 2207, 2213, 2233, 2239 \int_compare_p:nNn 1187, 1195, 1866, 1877, 1985 \int_eval:n 90 \int_incr:N 1653, 1693, 1695, 1709, 1711, 1715, 1717, 1806 \int_new:N 882, 883, 1232, 1233, 1234, 1235 | P \PackageError 7 |

| | | |
|--|---|--|
| \pagenumbering | 6 | 1008, 1010, 1063, 1069, 1086, 1092, |
| \pkg | 536, 546, 549 | 1115, 1146, 1153, 1191, 1207, 1223 |
| prg commands: | | |
| \prg_generate_conditional_-variant:Nnn | 146 | \sort_return_swapped: 27, 33, 921, 974, 1007, 1062, 1109, 1145, 1199, 1210, 1222 |
| \prg_new_conditional:Npnn | 140 | str commands: |
| \prg_return_false: | 144 | \str_case:nnTF 474, 514 |
| \prg_return_true: | 143 | \str_if_eq:nnTF 80 |
| \ProcessKeysOptions | 17, 668 | \str_if_eq_p:nn 1970, 1976, 1978, 1982 |
| prop commands: | | |
| \prop_get:NnN | 2153 | \str_new:N 417 |
| \prop_get:NnNTF | | \str_set:Nn 422, 424, 426, 428 |
| .. | 1888, 1915, 1920, 2137, 2182, 2192 | |
| \prop_if_exist:NTF | 673, 752 | |
| \prop_if_exist_p:N | 2144, 2188 | |
| \prop_if_in:NnTF | 25 | |
| \prop_if_in_p:Nn | 60, 2148 | |
| \prop_item:Nn | 27, 61 | |
| \prop_new:N | 167, 221, 626, 674, 753 | |
| \prop_put:Nnn | 165, 656, 732 | |
| \prop_remove:Nn | 164, 655, 727 | |
| \providecommand | 3 | |
| \ProvideDictionaryFor | | |
| | 8, 2267, 2430, 2579, 2724, 2883 | |
| \ProvideDictTranslation | 8, 156, 158 | |
| \ProvidesExplPackage | 14 | |
| R | | |
| \refstepcounter | 3 | |
| \RequirePackage | 16, 17, 18, 19, 20, 404, 667 | |
| S | | |
| \SaveTranslationFor | 148 | |
| seq commands: | | |
| \seq_clear:N | 357, 895 | |
| \seq_get_left:NN | 1294 | |
| \seq_if_empty:NTF | 1289 | |
| \seq_if_in:NnTF | 202, 942 | |
| \seq_map_break:n | 81, 1207, 1210 | |
| \seq_map_function:NN | 898 | |
| \seq_map_indexed_inline:Nn | 1167 | |
| \seq_map_inline:Nn | 1204 | |
| \seq_map_tokens:Nn | 63 | |
| \seq_new:N | 195, 342, 844, 892, 1227 | |
| \seq_pop_left:NN | 1288 | |
| \seq_put_right:Nn | 204, 944 | |
| \seq_reverse:N | 351 | |
| \seq_set_eq:NN | 1264 | |
| \seq_set_from_clist:Nn | 347, 850 | |
| \seq_sort:Nn | 901 | |
| sort commands: | | |
| \sort_return_same: | | |
| | 27, 33, 908, 913, 965, | |
| T | | |
| TeX and L ^A T _E X 2 _ε commands: | | |
| \@Alph | 55 | |
| \@addtoreset | 4 | |
| \@chapapp | 55 | |
| \@currentcounter | | |
| | 4, 21, 25, 28, 30, 33, 84, 86 | |
| \@currentlabel | 3 | |
| \@ifl@t@r | 3 | |
| \@ifpackageloaded | | |
| | 264, 284, 401, 541, 564, 602 | |
| \@onlypreamble | 159, 160 | |
| \@trns@lt@current@language | 538 | |
| \@trns@lt@language | 493, 554, 570 | |
| \bbl@loaded | 544 | |
| \bbl@main@language | 543 | |
| \c@ | 3 | |
| \c@page | 6, 94 | |
| \cl@ | 4 | |
| \hyper@link | 1573, 1821, 2003, 2087 | |
| \p@... | 3 | |
| \xpg@loaded | 567 | |
| \xpg@main@language | 566 | |
| \zref@addprop | 22, 32, 34, 36, 87, 88, 99 | |
| \zref@default | | |
| | 45, 1851, 1995, 2056, 2062, 2129 | |
| \zref@extractdefault | | |
| | 45, 939, 954, 956, 1000, 1001, 1004, 1006, 1018, 1022, 1026, 1030, 1054, 1055, 1059, 1061, 1081, 1104, 1219, 1221, 1304, 1309, 1578, 1583, 1589, 1824, 1825, 1827, 1830, 1844, 2008, 2012, 2017, 2032, 2092, 2096, 2101, 2106, 2121, 2204, 2205, 2208, 2210, 2214, 2216, 2226, 2227, 2230, 2231, 2234, 2236, 2240, 2242 | |
| \zref@ifpropundefined | 12 | |
| \zref@ifrefcontainsprop | 12, 1575, 1810, 1823, 1999, 2005, 2072, 2089 | |
| \zref@ifrefundefined | | |
| .. | 903, 905, 917, 1321, 1323, 1328, 1361, 1520, 1529, 1671, 1856, 1994 | |

| | |
|------------------------|--|
| \ZREF@mainlist | 22, 32, 34, 36, 87, 88, 99 |
| \zref@newprop | 4, 21, 23, 33, 35, 83, 85, 98 |
| \zref@refused | 1360 |
| \zref@wrapper@babel | 24, 843 |
| \textendash | 2260 |
| \texttt | 862 |
| \the | 3 |
| \thechapter | 55 |
| \thepage | 6, 95 |
| \thesection | 55 |
| tl commands: | |
| \c_empty_tl | 939, 954, 956, 1018, 1022, 1026, 1030, 1305, 1310 |
| \c_novalue_tl | 651, 722 |
| \tl_clear:N | 742, 750, 1036, 1037, 1265, 1266, 1267, 1268, 1269, 1291, 1648, 1649, 1650, 1651, 1692, 1857, 1860, 1884, 1908, 1955, 2194, 2196 |
| \tl_gset:Nn | 95 |
| \tl_head:N | 1041, 1043, 1126, 1128, 1142, 1144 |
| \tl_if_empty:NTF | 71, 771, 801, 830, 940, 1364, 1518, 1871, 1886, 2060 |
| \tl_if_empty:nTF | 163, 749, 1470, 1485, 1500, 1741, 1766, 1777, 1790, 1859 |
| \tl_if_empty_p:N | 962, 963, 971, 972, 979, 980, 1335, 1336, 1343, 1344, 1969, 1979, 1983, 2142, 2186 |
| \tl_if_empty_p:n | 1049, 1050, 1077, 1100 |
| \tl_if_eq:NNTF | 986, 1123, 1347 |
| \tl_if_eq:NnTF | 1177, 1180, 1206, 1209 |
| \tl_if_eq:nnTF | 999, 1053, 1169, 2203, 2225, 2229 |
| \tl_if_in:NnTF | 1082, 1105 |
| \tl_if_novalue:nTF | 654, 725 |
| \tl_map_break:n | 81 |
| \tl_map_tokens:Nn | 73 |
| \tl_new:N | 89, 245, 464, 465, 466, 587, 623, 669, 670, 884, 885, 886, 887, 888, 889, 890, 891, 1228, 1229, 1230, 1231, 1236, 1240, 1241, 1242, 1244, 1245, 1246, 1247, 1248, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1260, 1261 |
| \tl_put_left:Nn | 1552, 1559, 1608 |
| \tl_put_right:Nn | 1420, 1436, 1445, 1472, 1482, 1497, 1728, 1739, 1764, 1775, 1788, 1872, 1873, 1882 |
| \tl_reverse_items:n | 950, 950, 1020, 1024, 1028, 1032 |
| \tl_set:Nn | 252, 257, 266, 271, 290, 301, 489, 528, 580, 675, 741, 754, 938, 953, 955, |
| | 1017, 1019, 1021, 1023, 1025, 1027, 1029, 1031, 1040, 1042, 1080, 1103, 1130, 1132, 1134, 1136, 1298, 1299, 1302, 1307, 1411, 1412, 1535, 1566, 1684, 1685, 1708, 1868, 1869, 1881 |
| \tl_set_eq:NN | 478, 484, 518, 524, 538, 543, 566, 1646 |
| \tl_tail:N | 1131, 1133, 1135, 1137 |
| \l_tmpa_tl | 1036, 1040, 1049, 1077, 1080, 1083, 1123 |
| \l_tmppb_tl | 1037, 1042, 1050, 1100, 1103, 1106, 1123 |
| U | |
| use commands: | |
| \use:N | 21 |
| \UseHook | 585 |
| Z | |
| \zcDeclareTranslations | 20, 22, 103, 739 |
| \zcDicDefaultTransl | 8, 155, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2726, 2727, 2728, 2729, 2730, 2731, 2732, 2733, 2734, 2885, 2886, 2887, 2888, 2889, 2890, 2891, 2892, 2893 |
| \zcDicTypeTransl | 8, 155, 2283, 2284, 2285, 2286, 2288, 2289, 2290, 2291, 2293, 2294, 2295, 2296, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2307, 2308, 2309, 2310, 2312, 2313, 2314, 2315, 2316, 2317, 2319, 2320, 2321, 2322, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2333, 2334, 2335, 2336, 2338, 2339, 2340, 2341, 2343, 2344, 2345, 2346, 2348, 2349, 2350, 2351, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2364, 2365, 2366, 2367, 2369, 2370, 2371, 2372, 2374, 2375, 2376, 2377, 2379, 2380, 2381, 2382, 2384, 2385, 2386, 2387, 2389, 2390, 2391, 2392, 2394, 2395, 2396, 2397, 2399, 2400, 2401, 2402, 2404, 2405, 2406, 2407, 2409, 2410, 2411, 2412, 2414, 2415, 2416, 2417, 2419, 2420, 2421, 2422, 2424, 2425, 2426, 2427, 2442, 2443, 2444, 2445, 2447, 2448, 2449, 2450, 2452, 2453, 2454, 2455, 2457, 2458, 2459, 2460, 2462, 2463, 2464, 2465, 2467, 2468. |

| | |
|-------------------------------------|--|
| 2469, 2470, 2472, 2473, 2474, 2475, | 2907, 2908, 2910, 2911, 2912, 2913, |
| 2477, 2478, 2479, 2480, 2481, 2482, | 2915, 2916, 2917, 2918, 2920, 2921, |
| 2483, 2484, 2486, 2487, 2488, 2489, | 2922, 2923, 2925, 2926, 2927, 2928, |
| 2491, 2492, 2493, 2494, 2496, 2497, | 2930, 2931, 2932, 2933, 2935, 2936, |
| 2498, 2499, 2501, 2502, 2503, 2504, | 2937, 2938, 2940, 2941, 2942, 2943, |
| 2506, 2507, 2508, 2509, 2510, 2511, | 2945, 2946, 2947, 2948, 2950, 2951, |
| 2513, 2514, 2515, 2516, 2518, 2519, | 2952, 2953, 2955, 2956, 2957, 2958, |
| 2520, 2521, 2523, 2524, 2525, 2526, | 2959, 2960, 2962, 2963, 2964, 2965, |
| 2528, 2529, 2530, 2531, 2533, 2534, | 2967, 2968, 2969, 2970, 2972, 2973, |
| 2535, 2536, 2538, 2539, 2540, 2541, | 2974, 2975, 2977, 2978, 2979, 2980, |
| 2543, 2544, 2545, 2546, 2548, 2549, | 2982, 2983, 2984, 2985, 2987, 2988, |
| 2550, 2551, 2553, 2554, 2555, 2556, | 2989, 2990, 2992, 2993, 2994, 2995, |
| 2558, 2559, 2560, 2561, 2563, 2564, | 2997, 2998, 2999, 3000, 3002, 3003, |
| 2565, 2566, 2568, 2569, 2570, 2571, | 3004, 3005, 3007, 3008, 3009, 3010, |
| 2573, 2574, 2575, 2576, 2591, 2592, | 3012, 3013, 3014, 3015, 3017, 3018, |
| 2593, 2594, 2596, 2597, 2598, 2599, | 3019, 3020, 3022, 3023, 3024, 3025 |
| 2601, 2602, 2603, 2604, 2606, 2607, | \zcheck 121 |
| 2608, 2609, 2611, 2612, 2613, 2614, | \zcpageref 25, 876 |
| 2616, 2617, 2618, 2619, 2621, 2622, | \zceref 24, 26, 33, 35, 842, 879, 880 |
| 2623, 2624, 2626, 2627, 2628, 2629, | \zcRefTypeSetup 20, 21, 109, 671 |
| 2631, 2632, 2633, 2634, 2636, 2637, | \zcsetup 20, 665 |
| 2638, 2639, 2641, 2642, 2643, 2644, | zrefcheck commands: |
| 2646, 2647, 2648, 2649, 2651, 2652, | \zrefcheck_zceref_beg_label: ... 856 |
| 2653, 2654, 2655, 2656, 2658, 2659, | \zrefcheck_zceref_end_label_- |
| 2660, 2661, 2663, 2664, 2665, 2666, | maybe: 870 |
| 2668, 2669, 2670, 2671, 2673, 2674, | \zrefcheck_zceref_run_checks_on_- |
| 2675, 2676, 2678, 2679, 2680, 2681, | labels:n 871 |
| 2683, 2684, 2685, 2686, 2688, 2689, | zrefclever internal commands: |
| 2690, 2691, 2693, 2694, 2695, 2696, | \l__zrefclever_abbrev_bool 450, 454, 1875 |
| 2698, 2699, 2700, 2701, 2703, 2704, | \l__zrefclever_capitalize_bool .. 432, 436, 1863 |
| 2705, 2706, 2708, 2709, 2710, 2711, | \l__zrefclever_capitalize_first_- |
| 2713, 2714, 2715, 2716, 2718, 2719, | bool 433, 442, 1865 |
| 2720, 2721, 2736, 2737, 2738, 2739, | __zrefclever_counter_reset_by:n |
| 2741, 2742, 2743, 2744, 2746, 2747, | 5, 10, 39, 41, 43, 48, 50, 52, 57 |
| 2748, 2749, 2751, 2752, 2753, 2754, | __zrefclever_counter_reset_by_- |
| 2755, 2756, 2757, 2758, 2760, 2761, | aux:nn 64, 67 |
| 2762, 2763, 2765, 2766, 2767, 2768, | __zrefclever_counter_reset_by_- |
| 2769, 2770, 2772, 2773, 2774, 2775, | auxi:nnn 74, 78 |
| 2777, 2778, 2779, 2780, 2781, 2782, | \l__zrefclever_counter_resetby_- |
| 2783, 2784, 2786, 2787, 2788, 2789, | prop 4, 10, 60, 61, 221, 233 |
| 2791, 2792, 2793, 2794, 2796, 2797, | \l__zrefclever_counter_resettters_- |
| 2798, 2799, 2801, 2802, 2803, 2804, | seq 4, 10, 63, 195, 202, 205 |
| 2806, 2807, 2808, 2809, 2810, 2811, | \l__zrefclever_counter_type_prop |
| 2812, 2813, 2814, 2815, 2817, 2818, | 3, 9, 25, 28, 167, 179 |
| 2819, 2820, 2822, 2823, 2824, 2825, | \l__zrefclever_current_language_- |
| 2827, 2828, 2829, 2830, 2832, 2833, | tl 466, 485, 525, 538 |
| 2834, 2835, 2837, 2838, 2839, 2840, | __zrefclever_declare_default_- |
| 2842, 2843, 2844, 2845, 2847, 2848, | transl:nnn 8, 153, 2252, |
| 2849, 2850, 2852, 2853, 2854, 2855, | 2253, 2254, 2255, 2256, 2257, 2258, |
| 2857, 2858, 2859, 2860, 2862, 2863, | 2259, 2260, 2261, 2262, 2263, 2264 |
| 2864, 2865, 2867, 2868, 2869, 2870, | __zrefclever_declare_transl:nnn |
| 2872, 2873, 2874, 2875, 2877, 2878, | 8, 150, 773, 803, 807, 836 |
| 2879, 2880, 2895, 2896, 2897, 2898, | |
| 2900, 2901, 2902, 2903, 2905, 2906, | |

`__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` ... [1375](#), [1376](#), [1377](#), [2179](#)
`__zrefclever_get_option_with-`
`transl:nN` [55](#), [1276](#), [1277](#),
[1278](#), [1279](#), [1378](#), [1379](#), [1380](#), [1381](#),
[1382](#), [1383](#), [1384](#), [1385](#), [1386](#), [2133](#)
`__zrefclever_get_ref:n` [1423](#), [1439](#),
[1451](#), [1455](#), [1475](#), [1488](#), [1491](#), [1503](#),
[1506](#), [1540](#), [1560](#), [1731](#), [1744](#), [1749](#),
[1769](#), [1780](#), [1783](#), [1793](#), [1796](#), [1808](#)
`__zrefclever_get_ref_first:` ...
..... [35](#), [45](#), [1553](#), [1609](#), [1992](#)
`__zrefclever_get_transl:nnn` ...
. [8](#), [147](#), [1900](#), [1932](#), [1947](#), [2162](#), [2172](#)
`__zrefclever_if_transl:nn` . [140](#), [146](#)
`__zrefclever_if_transl:nnTF` ...
..... [7](#), [140](#), [1893](#), [1925](#), [1940](#), [2158](#)
`\l__zrefclever_label_a_tl`
..... [884](#), [1288](#), [1305](#), [1321](#), [1360](#),
[1361](#), [1367](#), [1411](#), [1423](#), [1439](#), [1455](#),
[1491](#), [1506](#), [1533](#), [1540](#), [1671](#), [1675](#),
[1684](#), [1708](#), [1731](#), [1749](#), [1783](#), [1796](#)
`\l__zrefclever_label_b_tl`
[884](#), [1291](#), [1294](#), [1310](#), [1323](#), [1328](#), [1675](#)
`\l__zrefclever_label_count_int` ..
..... [34](#), [1232](#),
[1270](#), [1373](#), [1405](#), [1652](#), [1680](#), [1806](#)
`\l__zrefclever_label_enclcnt_a-`
`tl` [884](#), [1017](#),
[1019](#), [1020](#), [1041](#), [1106](#), [1130](#), [1131](#)
`\l__zrefclever_label_enclcnt_b-`
`tl` [884](#), [1021](#),
[1023](#), [1024](#), [1043](#), [1083](#), [1132](#), [1133](#)
`\l__zrefclever_label_enclval_a-`
`tl` [884](#), [1025](#),
[1027](#), [1028](#), [1126](#), [1134](#), [1135](#), [1142](#)
`\l__zrefclever_label_enclval_b-`
`tl` [884](#), [1029](#),
[1031](#), [1032](#), [1128](#), [1136](#), [1137](#), [1144](#)
`\l__zrefclever_label_type_a_tl` ..
..... [884](#), [938](#), [940](#), [942](#),
[945](#), [953](#), [962](#), [971](#), [979](#), [986](#), [1177](#),
[1206](#), [1298](#), [1302](#), [1335](#), [1343](#), [1348](#),
[1364](#), [1412](#), [1685](#), [2142](#), [2145](#), [2149](#),
[2154](#), [2160](#), [2164](#), [2186](#), [2189](#), [2193](#)
`\l__zrefclever_label_type_b_tl` ..
[884](#), [955](#), [963](#), [972](#), [980](#), [986](#), [1180](#),
[1209](#), [1299](#), [1307](#), [1336](#), [1344](#), [1348](#)
`__zrefclever_label_type_put-`
`new_right:n` [27](#), [899](#), [936](#)

`\l__zrefclever_label_types_seq` ..
..... [27](#), [892](#), [895](#), [942](#), [945](#), [1204](#)
`__zrefclever_labels_in_sequence:nn`
..... [1532](#), [1674](#), [2199](#)
`\l__zrefclever_last_of_type_bool`
..... [33](#), [1225](#), [1319](#), [1324](#), [1325](#),
[1329](#), [1338](#), [1349](#), [1350](#), [1353](#), [1391](#)
`\l__zrefclever_lastsep_tl` . [1248](#),
[1382](#), [1438](#), [1454](#), [1474](#), [1490](#), [1502](#)
`\l__zrefclever_link_star_bool` ...
..... [844](#), [851](#), [1813](#), [1968](#), [2078](#)
`\l__zrefclever_listsep_tl`
... [1247](#), [1381](#), [1450](#), [1487](#), [1730](#),
[1743](#), [1748](#), [1768](#), [1779](#), [1782](#), [1792](#)
`\l__zrefclever_main_language_tl` .
..... [465](#), [479](#), [519](#), [543](#), [566](#), [580](#)
`\l__zrefclever_name_format-`
`fallback_tl` [1258](#),
[1881](#), [1884](#), [1886](#), [1922](#), [1944](#), [1951](#)
`\l__zrefclever_name_format_tl` ...
..... [1258](#),
[1868](#), [1869](#), [1872](#), [1873](#), [1881](#), [1882](#),
[1890](#), [1897](#), [1904](#), [1917](#), [1929](#), [1936](#)
`\l__zrefclever_name_in_link_bool`
.. [1258](#), [1568](#), [1972](#), [1988](#), [1989](#), [1997](#)
`\l__zrefclever_namefont_tl` [1240](#),
[1375](#), [1571](#), [1597](#), [2022](#), [2052](#), [2067](#)
`\l__zrefclever_nameinlink_str` ...
..... [417](#), [422](#),
[424](#), [426](#), [428](#), [1970](#), [1976](#), [1978](#), [1982](#)
`\l__zrefclever_nameinlink_tl` .. [417](#)
`\l__zrefclever_namesep_tl`
.. [1244](#), [1378](#), [2025](#), [2055](#), [2063](#), [2070](#)
`\l__zrefclever_next_is_same_bool`
..... [34](#), [53](#), [1234](#),
[1664](#), [1694](#), [1710](#), [1716](#), [2219](#), [2245](#)
`\l__zrefclever_next_maybe_range-`
`bool`
.. [34](#), [53](#), [1234](#), [1528](#), [1537](#), [1663](#),
[1690](#), [1701](#), [2211](#), [2218](#), [2237](#), [2244](#)
`\l__zrefclever_noabbrev_first-`
`bool` [451](#), [460](#), [1878](#)
`\l__zrefclever_notesept_tl`
..... [863](#), [1253](#), [1279](#)
`__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](#), [896](#), [929](#), [1296](#), [2201](#)
`\l__zrefclever_pairsep_tl`
..... [1246](#), [1380](#), [1422](#), [1538](#)

`__zrefclever_prop_put_non-`
`empty:Nnn` [9](#), [161](#), [178](#), [232](#)
`\l__zrefclever_range_beg_label-`
`tl` [34](#), [1234](#), [1269](#),
[1451](#), [1470](#), [1475](#), [1485](#), [1488](#), [1500](#),
[1503](#), [1651](#), [1692](#), [1708](#), [1741](#), [1744](#),
[1766](#), [1769](#), [1777](#), [1780](#), [1790](#), [1793](#)
`\l__zrefclever_range_count_int` ..
..... [34](#),
[1234](#), [1272](#), [1431](#), [1463](#), [1654](#), [1693](#),
[1705](#), [1709](#), [1715](#), [1723](#), [1760](#), [1801](#)
`\l__zrefclever_range_inhibit-`
`next_bool` [33](#), [34](#), [1234](#), [1669](#)
`\l__zrefclever_range_same_count-`
`int` [34](#),
[1234](#), [1273](#), [1418](#), [1452](#), [1463](#), [1655](#),
[1695](#), [1711](#), [1717](#), [1746](#), [1760](#), [1802](#)
`\l__zrefclever_rangesep_tl`
..... [1245](#), [1379](#), [1505](#), [1539](#), [1795](#)
`\l__zrefclever_ref_language_tl` ..
..... [464](#), [479](#), [485](#), [489](#),
[519](#), [525](#), [528](#), [1894](#), [1901](#), [1926](#),
[1933](#), [1941](#), [1948](#), [2159](#), [2163](#), [2173](#)
`\l__zrefclever_ref_options_prop` ..
..... [21](#), [626](#), [655](#), [656](#), [2137](#), [2182](#)
`\l__zrefclever_ref_property_tl` ..
..... [12](#), [245](#), [252](#),
[257](#), [266](#), [271](#), [290](#), [301](#), [1810](#), [1830](#),
[1844](#), [2000](#), [2033](#), [2073](#), [2107](#), [2122](#)
`\l__zrefclever_ref_typeset_font-`
`tl` [623](#), [625](#), [1282](#)
`\l__zrefclever_reffont_in_tl` [1242](#),
[1377](#), [1819](#), [1842](#), [2030](#), [2085](#), [2119](#)
`\l__zrefclever_reffont_out_tl` ...
..... [1241](#), [1376](#),
[1816](#), [1839](#), [2027](#), [2046](#), [2082](#), [2116](#)
`\l__zrefclever_refpos_in_tl` [1257](#),
[1386](#), [1831](#), [1845](#), [2034](#), [2108](#), [2123](#)
`\l__zrefclever_refpos_out_tl` [1255](#),
[1384](#), [1834](#), [1847](#), [2047](#), [2111](#), [2125](#)
`\l__zrefclever_refpre_in_tl` [1256](#),
[1385](#), [1829](#), [1843](#), [2031](#), [2105](#), [2120](#)
`\l__zrefclever_refpre_out_tl` [1254](#),
[1383](#), [1817](#), [1840](#), [2028](#), [2083](#), [2117](#)
`\l__zrefclever_setup_language_tl`
..... [669](#), [741](#), [773](#), [803](#), [807](#), [836](#)
`\l__zrefclever_setup_type_tl` ...
..... [669](#), [675](#), [728](#), [733](#),
[742](#), [750](#), [754](#), [771](#), [801](#), [808](#), [830](#), [837](#)
`\l__zrefclever_sort_decided_bool`
..... [949](#), [1034](#), [1038](#), [1057](#), [1068](#),
[1085](#), [1091](#), [1108](#), [1114](#), [1140](#), [1152](#)
`__zrefclever_sort_default:nn` ...
..... [26](#), [27](#), [931](#), [951](#)
`__zrefclever_sort_default-`
`different_types:nn` [993](#), [1160](#)
`__zrefclever_sort_default_same-`
`type:nn` [989](#), [1015](#)
`__zrefclever_sort_labels:`
..... [27](#), [33](#), [860](#), [893](#)
`__zrefclever_sort_page:nn`
..... [33](#), [930](#), [1216](#)
`\l__zrefclever_sort_prior_a_int` ..
..... [882](#), [1162](#), [1171](#), [1172](#), [1178](#), [1188](#), [1196](#)
`\l__zrefclever_sort_prior_b_int` ..
..... [883](#), [1163](#), [1173](#), [1174](#), [1181](#), [1189](#), [1197](#)
`\l__zrefclever_tlastsep_tl`
..... [1252](#), [1278](#), [1640](#)
`\l__zrefclever_tlistsep_tl`
..... [1251](#), [1277](#), [1618](#)
`\l__zrefclever_tpairsep_tl`
..... [1250](#), [1276](#), [1634](#)
`\l__zrefclever_type_<type>-`
`options_prop` [21](#)
`\l__zrefclever_type_count_int` ...
..... [34](#), [1232](#), [1271](#), [1615](#),
[1617](#), [1626](#), [1653](#), [1866](#), [1877](#), [1985](#)
`\l__zrefclever_type_first_label-`
`tl` [1227](#), [1267](#), [1411](#), [1520](#),
[1529](#), [1533](#), [1560](#), [1576](#), [1579](#), [1584](#),
[1590](#), [1649](#), [1684](#), [1856](#), [1994](#), [2000](#),
[2006](#), [2008](#), [2012](#), [2017](#), [2032](#), [2073](#),
[2090](#), [2092](#), [2096](#), [2101](#), [2106](#), [2121](#)
`\l__zrefclever_type_first_label-`
`type_tl` [1227](#),
[1268](#), [1412](#), [1524](#), [1650](#), [1685](#), [1859](#),
[1889](#), [1896](#), [1903](#), [1910](#), [1916](#),
[1921](#), [1928](#), [1935](#), [1943](#), [1950](#), [1957](#)
`__zrefclever_type_name_setup:` ..
..... [35](#), [1548](#), [1854](#)
`\l__zrefclever_type_name_tl` . [46](#),
[1258](#), [1592](#), [1598](#), [1857](#), [1860](#), [1891](#),
[1900](#), [1908](#), [1918](#), [1923](#), [1932](#), [1947](#),
[1955](#), [1969](#), [2023](#), [2053](#), [2060](#), [2068](#)
`\l__zrefclever_typeset_compress-`
`bool` [360](#), [363](#), [1666](#)
`\l__zrefclever_typeset_labels-`
`seq` ... [1227](#), [1264](#), [1288](#), [1289](#), [1294](#)
`\l__zrefclever_typeset_last_bool`
..... [33](#), [1225](#),
[1285](#), [1286](#), [1292](#), [1318](#), [1623](#), [1984](#)
`\l__zrefclever_typeset_name_bool`
..... [308](#), [315](#), [320](#), [325](#), [1550](#), [1564](#)
`\l__zrefclever_typeset_queue-`
`curr_tl` [1227](#), [1266](#), [1420](#),
[1436](#), [1445](#), [1472](#), [1482](#), [1497](#), [1518](#),
[1535](#), [1552](#), [1559](#), [1566](#), [1609](#), [1630](#),

| | |
|--|---|
| 1635, 1641, 1647, 1648, 1728, 1739, 1764, 1775, 1788, 1871, 1979, 1983 | \l__zrefclever_typesort_seq |
| \l__zrefclever_typeset_queue_- prev_tl 1227, 1265, 1619, 1647 | 342, 347, 351, 357, 1167 |
| \l__zrefclever_typeset_range_- bool 369, 372, 859, 1516 | \l__zrefclever_use_hyperref_bool 376, 403, 409, 1813, 1967, 2077 |
| \l__zrefclever_typeset_ref_bool 307, 314, 319, 324, 1550, 1557 | \l__zrefclever_warn_hyperref_- bool 376, 407 |
| __zrefclever_typeset_refs: 33, 34, 45, 46, 49, 861, 1262 | __zrefclever_zcref:nnn 843, 846 |
| __zrefclever_typeset_refs_aux_- last_of_type: 1394, 1402 | __zrefclever_zcref:nnnn . 24, 26, 846 |
| __zrefclever_typeset_refs_aux_- not_last_of_type: 1398, 1658 | \l__zrefclever_zcref_labels_seq 26, 844, 850, 872, 899, 901, 1264 |
| \l__zrefclever_typeset_sort_bool 333, 336, 858 | \l__zrefclever_zcref_note_tl 587, 590, 864 |
| | \l__zrefclever_zcref_with_check_- bool 594, 609, 855, 868 |
| | \l__zrefclever_zrefcheck_- available_bool 593, 604, 615, 854, 867 |