

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 `\c1@<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 `\c1@<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 `\c1@<counter>` cannot possibly fully account for all of the automatic counter resetting which takes place in the document. And there’s also no other “general rule” we could grab on for this, as far as I know. So we provide a way to manually tell `zref-clever` of these cases, by means of the `counterresetby` option, whose information is stored in `\l__zrefclever_counter_resetby_prop`. This manual specification has precedence over the search through `\l__zrefclever_counter_resettters_seq`, and should be handled with care, since there is no possible verification mechanism for this.

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

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

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

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

```

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

```

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

`\l__zrefclever_ref_property_tl` stores the property to which the reference is being made. Currently, we restrict `ref=` to these two (or three) alternatives – `zc@thecnt`, `page`, and `title` if `zref-titleref` is loaded –, 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.

```

245 \tl_new:N \l__zrefclever_ref_property_tl
246 \keys_define:nn { zref-clever / reference }
247 {
248     ref .choice: ,
249     ref / zc@thecnt .code:n =
250     { \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt } } ,
251     ref / page .code:n =
252     { \tl_set:Nn \l__zrefclever_ref_property_tl { page } } ,
253     ref / title .code:n =
254     {
255         \AddToHook { begindocument }
256         {
257             \@ifpackageloaded { zref-titleref }
258             { \tl_set:Nn \l__zrefclever_ref_property_tl { title } }
259             {
260                 \msg_warning:nn { zref-clever } { missing-zref-titleref }
261                 \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
262             }
263         }
264     }

```

```

264     } ,
265     ref .initial:n = zc@thecnt ,
266     ref .value_required:n = true ,
267     page .meta:n = { ref = page },
268     page .value_forbidden:n = true ,
269 }
270 \AddToHook { begindocument }
271 {
272   \@ifpackageloaded { zref-titleref }
273   {
274     \keys_define:nn { zref-clever / reference }
275     {
276       ref / title .code:n =
277       { \tl_set:Nn \l__zrefclever_ref_property_tl { title } }
278     }
279   }
280   {
281     \keys_define:nn { zref-clever / reference }
282     {
283       ref / title .code:n =
284       {
285         \msg_warning:nn { zref-clever } { missing-zref-titleref }
286         \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
287       }
288     }
289   }
290 }

```

typeset option

```

291 \bool_new:N \l__zrefclever_typeset_ref_bool
292 \bool_new:N \l__zrefclever_typeset_name_bool
293 \keys_define:nn { zref-clever / reference }
294 {
295   typeset .choice: ,
296   typeset / both .code:n =
297   {
298     \bool_set_true:N \l__zrefclever_typeset_ref_bool
299     \bool_set_true:N \l__zrefclever_typeset_name_bool
300   } ,
301   typeset / ref .code:n =
302   {
303     \bool_set_true:N \l__zrefclever_typeset_ref_bool
304     \bool_set_false:N \l__zrefclever_typeset_name_bool
305   } ,
306   typeset / name .code:n =
307   {
308     \bool_set_false:N \l__zrefclever_typeset_ref_bool
309     \bool_set_true:N \l__zrefclever_typeset_name_bool
310   } ,
311   typeset .initial:n = both ,
312   typeset .value_required:n = true ,
313
314   noname .meta:n = { typeset = ref },

```

```

315     noname .value_forbidden:n = true ,
316 }

```

sort option

```

317 \bool_new:N \l__zrefclever_typeset_sort_bool
318 \keys_define:nn { zref-clever / reference }
319 {
320     sort .bool_set:N = \l__zrefclever_typeset_sort_bool ,
321     sort .initial:n = true ,
322     sort .default:n = true ,
323     nosort .meta:n = { sort = false },
324     nosort .value_forbidden:n = true ,
325 }

```

typesort option

`\l__zrefclever_typesort_seq` is stored reversed, since the sort priorities are computed in the negative range in `__zrefclever_sort_default_different_types:nn`, so that we can implicitly rely on ‘0’ being the “last value”, and spare creating an integer variable using `\seq_map_indexed_inline:Nn`.

```

326 \seq_new:N \l__zrefclever_typesort_seq
327 \keys_define:nn { zref-clever / reference }
328 {
329     typesort .code:n =
330     {
331         \seq_set_from_clist:Nn \l__zrefclever_typesort_seq {#1}
332         \seq_reverse:N \l__zrefclever_typesort_seq
333     } ,
334     typesort .initial:n =
335     { part , chapter , section , paragraph },
336     typesort .value_required:n = true ,
337     notypesort .code:n =
338     { \seq_clear:N \l__zrefclever_typesort_seq } ,
339     notypesort .value_forbidden:n = true ,
340 }

```

comp option

```

341 \bool_new:N \l__zrefclever_typeset_compress_bool
342 \keys_define:nn { zref-clever / reference }
343 {
344     comp .bool_set:N = \l__zrefclever_typeset_compress_bool ,
345     comp .initial:n = true ,
346     comp .default:n = true ,
347     nocomp .meta:n = { comp = false },
348     nocomp .value_forbidden:n = true ,
349 }

```

range option

```

350 \bool_new:N \l__zrefclever_typeset_range_bool
351 \keys_define:nn { zref-clever / reference }
352 {
353     range .bool_set:N = \l__zrefclever_typeset_range_bool ,
354     range .initial:n = false ,

```

```

355     range .default:n = true ,
356 }

```

hyperref option

```

357 \bool_new:N \l__zrefclever_use_hyperref_bool
358 \bool_new:N \l__zrefclever_warn_hyperref_bool
359 \keys_define:nn { zref-clever / reference }
360 {
361   hyperref .choice: ,
362   hyperref / auto .code:n =
363   {
364     \bool_set_true:N \l__zrefclever_use_hyperref_bool
365     \bool_set_false:N \l__zrefclever_warn_hyperref_bool
366   } ,
367   hyperref / true .code:n =
368   {
369     \bool_set_true:N \l__zrefclever_use_hyperref_bool
370     \bool_set_true:N \l__zrefclever_warn_hyperref_bool
371   } ,
372   hyperref / false .code:n =
373   {
374     \bool_set_false:N \l__zrefclever_use_hyperref_bool
375     \bool_set_false:N \l__zrefclever_warn_hyperref_bool
376   } ,
377   hyperref .initial:n = auto ,
378   hyperref .default:n = auto
379 }
380 \AddToHook { begindocument }
381 {
382   \@ifpackageloaded { hyperref }
383   {
384     \bool_if:NT \l__zrefclever_use_hyperref_bool
385     { \RequirePackage { zref-hyperref } }
386   }
387   {
388     \bool_if:NT \l__zrefclever_warn_hyperref_bool
389     { \msg_warning:nn { zref-clever } { missing-hyperref } }
390     \bool_set_false:N \l__zrefclever_use_hyperref_bool
391   }
392   \keys_define:nn { zref-clever / reference }
393   {
394     hyperref .code:n =
395     { \msg_warning:nn { zref-clever } { hyperref-preamble-only } }
396   }
397 }

```

nameinlink option

```

398 \str_new:N \l__zrefclever_nameinlink_str
399 \keys_define:nn { zref-clever / reference }
400 {
401   nameinlink .choice: ,
402   nameinlink / true .code:n =
403   { \str_set:Nn \l__zrefclever_nameinlink_str { true } } ,
404   nameinlink / false .code:n =

```

```

405     { \str_set:Nn \l__zrefclever_nameinlink_str { false } } ,
406     nameinlink / single .code:n =
407     { \str_set:Nn \l__zrefclever_nameinlink_str { single } } ,
408     nameinlink / tsingle .code:n =
409     { \str_set:Nn \l__zrefclever_nameinlink_str { tsingle } } ,
410     nameinlink .initial:n = tsingle ,
411     nameinlink .default:n = true ,
412 }

```

cap and capfirst options

```

413 \bool_new:N \l__zrefclever_capitalize_bool
414 \bool_new:N \l__zrefclever_capitalize_first_bool
415 \keys_define:nn { zref-clever / reference }
416 {
417     cap .bool_set:N = \l__zrefclever_capitalize_bool ,
418     cap .initial:n = false ,
419     cap .default:n = true ,
420     nocap .meta:n = { cap = false },
421     nocap .value_forbidden:n = true ,
422
423     capfirst .bool_set:N = \l__zrefclever_capitalize_first_bool ,
424     capfirst .initial:n = false ,
425     capfirst .default:n = true ,
426
427     C .meta:n =
428     { capfirst = true , noabbrevfirst = true },
429     C .value_forbidden:n = true ,
430 }

```

abbrev and noabbrevfirst options

```

431 \bool_new:N \l__zrefclever_abbrev_bool
432 \bool_new:N \l__zrefclever_noabbrev_first_bool
433 \keys_define:nn { zref-clever / reference }
434 {
435     abbrev .bool_set:N = \l__zrefclever_abbrev_bool ,
436     abbrev .initial:n = false ,
437     abbrev .default:n = true ,
438     noabbrev .meta:n = { abbrev = false },
439     noabbrev .value_forbidden:n = true ,
440
441     noabbrevfirst .bool_set:N = \l__zrefclever_noabbrev_first_bool ,
442     noabbrevfirst .initial:n = false ,
443     noabbrevfirst .default:n = true ,
444 }

```

lang option

`\l__zrefclever_current_language_tl` is an internal alias for translations’s internal macro `\@trnslt@current@language` which, in turn, is an alias for `\language` used by both `babel` and `polyglossia`, but translations ensures it always exists, even if no language package is loaded. `\l__zrefclever_main_language_tl` is an internal alias for `babel`’s `\bbl@main@language` or for `polyglossia`’s `\xpg@main@language`, as the case may be. `\l__zrefclever_ref_language_tl` is the internal variable which stores the language in which the reference is to be made.

The overall setup here seems a little roundabout, but this is actually required. In the preamble, we (potentially) don't yet have values for the “main” and “current” document languages, this must be retrieved at a `begindocument/before` hook. And it must be `before`, since `\LoadDictionaryFor` is preamble only. The `begindocument/before` hook is responsible to get values for `\l__zrefclever_main_language_tl` and `\l__zrefclever_current_language_tl` and load `zref-clever` dictionaries for all languages loaded by `babel` or `polyglossia`, or directly specified by the user. After this information is retrieved, the preamble options are executed, and this is handled by the internal `zref-clever/reflanguage` hook, which is called at this point. This hook handles two things: it executes the preamble options and, in sequence, it redefines the `lang` option key, since in the document body, we can handle “main” and “current” language options immediately. This redefinition is added to the `zref-clever/reflanguage` hook, but `\AtEndOfPackage` so that it comes after `\ProcessKeysOptions`. In other words, this is how we ensure the preamble options are executed before the `lang` key is redefined.

For the `babel` and `polyglossia` variables which store the “main” and “current” languages, see <https://tex.stackexchange.com/a/233178>, including comments, particularly the one by Javier Bezos. For the `babel` and `polyglossia` variables which store the list of loaded languages, see <https://tex.stackexchange.com/a/281220>, including comments, particularly PLK's.

```

445 \tl_new:N \l__zrefclever_ref_language_tl
446 \tl_new:N \l__zrefclever_main_language_tl
447 \tl_new:N \l__zrefclever_current_language_tl
448 \NewHook { zref-clever / reflanguage }
449 \keys_define:nn { zref-clever / reference }
450 {
451   lang .code:n =
452   {
453     \AddToHook { zref-clever / reflanguage }
454     {
455       \str_case:nnF {#1}
456       {
457         { main }
458         {
459           \tl_set_eq:NN \l__zrefclever_ref_language_tl
460             \l__zrefclever_main_language_tl
461         }
462
463         { current }
464         {
465           \tl_set_eq:NN \l__zrefclever_ref_language_tl
466             \l__zrefclever_current_language_tl
467         }
468       }
469     }
470     \tl_set:Nn \l__zrefclever_ref_language_tl {#1}

```

If the user specified a language in the preamble, make sure it is loaded. There's no need to worry with redundancy with `babel` and `polyglossia` loaded languages, since `\LoadDictionaryFor` does not reload a dictionary if it's already been loaded.

```

471 \exp_args:Nx \file_if_exist:nTF
472 { zref-clever- \@trnslt@language {#1} .trsl }
473 { \LoadDictionaryFor {#1} { zref-clever } }

```



```

474         {
475             \exp_args:Nx \file_if_exist:nT
476             { zref-clever- \baselanguage {#1} .trsl }
477             { \LoadDictionaryFor {#1} { zref-clever } }
478         }
479     }
480 }
481 },
482 lang .initial:n = main ,
483 lang .value_required:n = true ,
484 }

```

Redefinition of the `lang` key option for the document body.

```

485 \AtEndOfPackage
486 {
487     \AddToHook { zref-clever / reflanguage }
488     {
489         \keys_define:nn { zref-clever / reference }
490         {
491             lang .code:n =
492             {
493                 \str_case:nnF {#1}
494                 {
495                     { main }
496                     {
497                         \tl_set_eq:NN \l__zrefclever_ref_language_tl
498                         \l__zrefclever_main_language_tl
499                     }
500
501                     { current }
502                     {
503                         \tl_set_eq:NN \l__zrefclever_ref_language_tl
504                         \l__zrefclever_current_language_tl
505                     }
506                 }
507                 { \tl_set:Nn \l__zrefclever_ref_language_tl {#1} }
508             } ,
509             lang .value_required:n = true ,
510         }
511     }
512 }
513 \AddToHook { begindocument / before }
514 {
515     \tl_set_eq:NN \l__zrefclever_current_language_tl
516     \@trnslt@current@language
517     \@ifpackageloaded{babel}
518     {
519         \tl_set_eq:NN \l__zrefclever_main_language_tl
520         \bbl@main@language
521         \clist_map_inline:Nn \bbl@loaded
522         {

```

Funny enough, translations also loads its basic dictionaries for all languages loaded by babel or polyglossia. First, there is no way to disable this, even if we don't need them at

all here. Second, translations sends messages of its own missing dictionaries to **info** and everyone else's to **warning**... So we have to control ourselves for missing dictionaries and load them only if available.

```

523         \exp_args:Nx \file_if_exist:nTF
524         { zref-clever- \@trnslt@language {#1} .trsl }
525         { \LoadDictionaryFor {#1} { zref-clever } }
526         {
527             \exp_args:Nx \file_if_exist:nT
528             { zref-clever- \baselanguage {#1} .trsl }
529             { \LoadDictionaryFor {#1} { zref-clever } }
530         }
531     }
532 }
533 {
534     \@ifpackageloaded{polyglossia}
535     {
536         \tl_set_eq:NN \l__zrefclever_main_language_tl
537         \xpg@main@language
538         \clist_map_inline:Nn \xpg@loaded
539         {
540             \exp_args:Nx \file_if_exist:nTF
541             { zref-clever- \@trnslt@language {#1} .trsl }
542             { \LoadDictionaryFor {#1} { zref-clever } }
543             {
544                 \exp_args:Nx \file_if_exist:nT
545                 { zref-clever- \baselanguage {#1} .trsl }
546                 { \LoadDictionaryFor {#1} { zref-clever } }
547             }
548         }
549     }
550     {
551         \tl_set:Nn \l__zrefclever_main_language_tl { english }
552         \LoadDictionaryFor { english } { zref-clever }
553     }
554 }

```

Then we execute the package options stored in the zref-clever/reflanguage hook.

```

555     \UseHook { zref-clever / reflanguage }
556 }

```

note option

```

557 \tl_new:N \l__zrefclever_zceref_note_tl
558 \keys_define:nn { zref-clever / reference }
559 {
560     note .tl_set:N = \l__zrefclever_zceref_note_tl ,
561     note .value_required:n = true ,
562 }

```

check option

Integration with zref-check.

```

563 \bool_new:N \l__zrefclever_zrefcheck_available_bool
564 \bool_new:N \l__zrefclever_zceref_with_check_bool

```

```

565 \keys_define:nn { zref-clever / reference }
566 {
567   check .code:n =
568     { \msg_warning:nn { zref-clever } { check-document-only } } ,
569 }
570 \AddToHook { begindocument }
571 {
572   \@ifpackageloaded { zref-check }
573   {
574     \bool_set_true:N \l__zrefclever_zrefcheck_available_bool
575     \keys_define:nn { zref-clever / reference }
576     {
577       check .code:n =
578       {
579         \bool_set_true:N \l__zrefclever_zcref_with_check_bool
580         \keys_set:nn { zref-check / zcheck } {#1}
581       }
582     }
583   }
584   {
585     \bool_set_false:N \l__zrefclever_zrefcheck_available_bool
586     \keys_define:nn { zref-clever / reference }
587     {
588       check .code:n =
589       { \msg_warning:nn { zref-clever } { missing-zref-check } }
590     }
591   }
592 }

```

Reference options

```

593 \tl_new:N \l__zrefclever_ref_typeset_font_tl
594 \keys_define:nn { zref-clever / reference }
595 { font .tl_set:N = \l__zrefclever_ref_typeset_font_tl }

```

Only not necessarily type-specific options are pertinent here.

```

596 \prop_new:N \l__zrefclever_ref_options_prop
597 \clist_map_inline:nn
598 {
599   % Not type-specific options.
600   tpairsep ,
601   tlistsep ,
602   tlastsep ,
603   notesep ,
604   % Possibly type-specific options.
605   namefont ,
606   namesep ,
607   pairsep ,
608   listsep ,
609   lastsep ,
610   rangesep ,
611   reffont ,
612   refpre ,
613   refpos ,
614   reffont-in ,

```

```

615     refpre-in ,
616     refpos-in ,
617   }
618   {
619     \keys_define:nn { zref-clever / reference }
620     {
621       #1 .default:V = \c_novalue_tl ,
622       #1 .code:n =
623       {
624         \tl_if_novalue:nTF {##1}
625         { \prop_remove:Nn \l__zrefclever_ref_options_prop {#1} }
626         { \prop_put:Nnn \l__zrefclever_ref_options_prop {#1} {##1} }
627       } ,
628     }
629   }

```

Package options

```

630 \keys_define:nn { }
631 {
632   zref-clever / zcsetup .inherit:n = zref-clever / label ,
633   zref-clever / zcsetup .inherit:n = zref-clever / reference ,
634 }

```

`\zcsetup` Provide `\zcsetup`.

```

635 \NewDocumentCommand \zcsetup { m }
636 { \keys_set:nn { zref-clever / zcsetup } {#1} }

```

(*End definition for \zcsetup.*)

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

```

637 \RequirePackage { l3keys2e }
638 \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
639 \tl_new:N \l__zrefclever_setup_type_tl
640 \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`.

```

641 \NewDocumentCommand \zcRefTypeSetup { m m }
642 {
643   \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
644   { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
645   \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
646   \keys_set:nn { zref-clever / typesetup } {#2}
647 }

```

(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.

```

648 \clist_map_inline:nn
649 {
650     tpairsep ,
651     tlistsep ,
652     tlastsep ,
653     notesep ,
654 }
655 {
656     \keys_define:nn { zref-clever / typesetup }
657     {
658         #1 .code:n =
659         {
660             \msg_warning:nnn { zref-clever } { option-not-type-specific } {#1}
661         } ,
662     }
663 }
```

Possibly or necessarily type-specific options.

```

664 \clist_map_inline:nn
665 {
666     % Possibly type-specific options.
667     namefont ,
668     namesep ,
669     pairsep ,
670     listsep ,
671     lastsep ,
672     rangesep ,
673     reffont ,
674     refpre ,
675     refpos ,
676     reffont-in ,
677     refpre-in ,
678     refpos-in ,
679     % Necessarily type-specific options.
680     Name-sg ,
681     name-sg ,
682     Name-pl ,
683     name-pl ,
```

```

684     Name-sg-ab ,
685     name-sg-ab ,
686     Name-pl-ab ,
687     name-pl-ab ,
688   }
689   {
690     \keys_define:nn { zref-clever / typesetup }
691     {
692       #1 .default:V = \c_novalue_tl ,
693       #1 .code:n =
694       {
695         \tl_if_novalue:nTF {##1}
696         {
697           \prop_remove:cn
698             { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
699             {#1}
700         }
701         {
702           \prop_put:cnn
703             { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
704             {#1} {##1}
705         }
706       } ,
707     }
708   }

```

5.2 \zcDeclareTranslations

\zcDeclareTranslations Provide \zcDeclareTranslations.

```

709 \NewDocumentCommand \zcDeclareTranslations { m m }
710 {
711   \tl_set:Nn \l__zrefclever_setup_language_tl {#1}
712   \tl_clear:N \l__zrefclever_setup_type_tl
713   \keys_set:nn { zref-clever / translations } {#2}
714 }

```

(End definition for \zcDeclareTranslations.)

```

715 \keys_define:nn { zref-clever / translations }
716 {
717   type .code:n =
718   {
719     \tl_if_empty:nTF {#1}
720     { \tl_clear:N \l__zrefclever_setup_type_tl }
721     {
722       \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
723       { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
724       \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
725     }
726   } ,
727 }

```

Not type-specific options.

```

728 \clist_map_inline:nn
729 {

```

```

730     tpairsep ,
731     tlistsep ,
732     tlastsep ,
733     notesep ,
734 }
735 {
736   \keys_define:nn { zref-clever / translations }
737   {
738     #1 .value_required:n = true ,
739     #1 .code:n =
740     {
741       \tl_if_empty:NTF \l__zrefclever_setup_type_tl
742       {
743         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
744         { zrefclever-default- #1 } {##1}
745       }
746       {
747         \msg_warning:nnn { zref-clever }
748         { option-not-type-specific } {#1}
749       }
750     } ,
751   }
752 }

```

Possibly type-specific options.

```

753 \clist_map_inline:nn
754 {
755   namesep ,
756   pairsep ,
757   listsep ,
758   lastsep ,
759   rangesep ,
760   refpre ,
761   refpos ,
762   refpre-in ,
763   refpos-in ,
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         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
778         { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
779       }
780     } ,
781   }
782 }

```

Necessarily type-specific options.

```

783 \clist_map_inline:nn
784 {
785   Name-sg ,
786   name-sg ,
787   Name-pl ,
788   name-pl ,
789   Name-sg-ab ,
790   name-sg-ab ,
791   Name-pl-ab ,
792   name-pl-ab ,
793 }
794 {
795   \keys_define:nn { zref-clever / translations }
796   {
797     #1 .value_required:n = true ,
798     #1 .code:n =
799     {
800       \tl_if_empty:NTF \l__zrefclever_setup_type_tl
801       {
802         \msg_warning:nnn { zref-clever }
803         { option-only-type-specific } {#1}
804       }
805       {
806         \__zrefclever_declare_transl:xxn { \l__zrefclever_setup_language_tl }
807         { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
808       }
809     } ,
810   }
811 }

```

6 \zcref

```

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

```

(End definition for \zcref.)

```

\l__zrefclever_zcref_labels_seq
\l__zrefclever_link_star_bool
814 \seq_new:N \l__zrefclever_zcref_labels_seq
815 \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>}
```



```

816 \cs_new_protected:Npn \__zrefclever_zcref:nnn #1#2#3
817 {
818   \group_begin:
819     \keys_set:nn { zref-clever / reference } {#3}
820     \seq_set_from_clist:Nn \l__zrefclever_zcref_labels_seq {#1}
821     \bool_set:Nn \l__zrefclever_link_star_bool {#2}
822     % Integration with 'zref-check'.
823     \bool_lazy_and:nnT
824       { \l__zrefclever_zrefcheck_available_bool }
825       { \l__zrefclever_zcref_with_check_bool }
826       { \zrefcheck_zcref_beg_label: }
827     \bool_lazy_or:nnT
828       { \l__zrefclever_typeset_sort_bool }
829       { \l__zrefclever_typeset_range_bool }
830       { \__zrefclever_sort_labels: }
831     \__zrefclever_typeset_refs:
832     % Typeset \texttt{note}.
833     \l__zrefclever_noteseq_tl
834     \l__zrefclever_zcref_note_tl
835     % Integration with 'zref-check'.
836     \bool_lazy_and:nnT
837       { \l__zrefclever_zrefcheck_available_bool }
838       { \l__zrefclever_zcref_with_check_bool }
839       {
840         \zrefcheck_zcref_end_label_maybe:
841         \zrefcheck_zcref_run_checks_on_labels:n
842           { \l__zrefclever_zcref_labels_seq }
843       }
844   \group_end:
845 }

```

(End definition for __zrefclever_zcref:nnnn.)

7 \zcpageref

```

\zcpageref \zcpageref{*}[\langle options \rangle]{\langle labels \rangle}
846 \NewDocumentCommand \zcpageref { s O { } m }
847 {
848   \IfBooleanTF {#1}
849     { \zcref*[#2, ref = page] {#3} }
850     { \zcref [ #2, ref = page] {#3} }
851 }

```

(End definition for \zcpageref.)

8 Sorting

```

852 \int_new:N \l__zrefclever_sort_prior_a_int
853 \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
\l__zrefclever_label_b_tl
\l__zrefclever_label_type_a_tl
\l__zrefclever_label_type_b_tl
\l__zrefclever_label_enclcnt_a_tl
\l__zrefclever_label_enclcnt_b_tl
\l__zrefclever_label_enclval_a_tl
\l__zrefclever_label_enclval_b_tl

```

```

854 \tl_new:N \l__zrefclever_label_a_tl
855 \tl_new:N \l__zrefclever_label_b_tl
856 \tl_new:N \l__zrefclever_label_type_a_tl
857 \tl_new:N \l__zrefclever_label_type_b_tl
858 \tl_new:N \l__zrefclever_label_enclcnt_a_tl
859 \tl_new:N \l__zrefclever_label_enclcnt_b_tl
860 \tl_new:N \l__zrefclever_label_enclval_a_tl
861 \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`.

```

862 \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.

```

863 \cs_new_protected:Npn \__zrefclever_sort_labels:
864 {

```

Store label types sequence.

```

865   \seq_clear:N \l__zrefclever_label_types_seq
866   \tl_if_eq:NnF \l__zrefclever_ref_property_tl { page }
867   {
868     \seq_map_function:NN
869     \l__zrefclever_zcref_labels_seq \__zrefclever_label_type_put_new_right:n
870   }

```

Sort.

```

871   \seq_sort:Nn \l__zrefclever_zcref_labels_seq
872   {
873     \zref@ifrefundefined {##1}
874     {
875       \zref@ifrefundefined {##2}
876       {
877         % Neither label is defined.
878         \sort_return_same:
879       }
880       {
881         % The second label is defined, but the first isn't, leave the
882         % undefined first (to be more visible).
883         \sort_return_same:
884       }
885     }
886     {
887       \zref@ifrefundefined {##2}
888       {
889         % The first label is defined, but the second isn't, bring the
890         % second forward.

```

```

891         \sort_return_swapped:
892     }
893     {
894         % The interesting case: both labels are defined. The
895         % reference to the "default" property/counter or to the page
896         % are quite different from our perspective, they rely on
897         % different fields and even use different information for
898         % sorting, so we branch them here to specialized functions.
899         \tl_if_eq:NnTF \l__zrefclever_ref_property_tl { page }
900             { \__zrefclever_sort_page:nn {##1} {##2} }
901             { \__zrefclever_sort_default:nn {##1} {##2} }
902     }
903 }
904 }
905 }

```

(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>}
906 \cs_new_protected:Npn \__zrefclever_label_type_put_new_right:n #1
907 {
908     \tl_set:Nx \l__zrefclever_label_type_a_tl
909     { \zref@extractdefault {#1} {zc@type} { \c_empty_tl } }
910     \tl_if_empty:NF \l__zrefclever_label_type_a_tl
911     {
912         \seq_if_in:NVF \l__zrefclever_label_types_seq \l__zrefclever_label_type_a_tl
913         {
914             \seq_put_right:NV
915             \l__zrefclever_label_types_seq \l__zrefclever_label_type_a_tl
916         }
917     }
918 }

```

(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.

```

919 \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.

```

920 \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}}
921 \cs_new_protected:Npn \__zrefclever_sort_default:nn #1#2
922 {
923   \tl_set:Nx \l__zrefclever_label_type_a_tl
924   { \zref@extractdefault {#1} {zc@type} { \c_empty_tl } }
925   \tl_set:Nx \l__zrefclever_label_type_b_tl
926   { \zref@extractdefault {#2} {zc@type} { \c_empty_tl } }
927
928   \bool_if:nTF
929   {
930     % The second label has a type, but the first doesn't, leave the
931     % undefined first (to be more visible).
932     \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
933     ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
934   }
935   { \sort_return_same: }
936   {
937     \bool_if:nTF
938     {
939       % The first label has a type, but the second doesn't, bring the
940       % second forward.
941       ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
942       \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
943     }
944     { \sort_return_swapped: }
945     {
946       \bool_if:nTF
947       {
948         % The interesting case: both labels have a type\dots{}
949         ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
950         ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
951       }
952       {
953         % Here we send this to a couple of auxiliary functions for no
954         % other reason than to keep this long function a little less
955         % unreadable.
956         \tl_if_eq:NNTF \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
957         {
958           % \dots{} and it's the same type.
959           \__zrefclever_sort_default_same_type:nn {#1} {#2}
960         }
961         {
962           % \dots{} and they are different types.
963           \__zrefclever_sort_default_different_types:nn {#1} {#2}
964         }
965       }
966     }
967     {
968       % Neither of the labels has a type. We can't do much of
969       % meaningful here, but if it's the same counter, compare it.
970       \exp_args:Nxx \tl_if_eq:nnTF
971       { \zref@extractdefault {#1} { counter } { } }
972       { \zref@extractdefault {#2} { counter } { } }
973       {

```

```

973         \int_compare:nNnTF
974         { \zref@extractdefault {#1} { zc@cntval } {-1} }
975         >
976         { \zref@extractdefault {#2} { zc@cntval } {-1} }
977         { \sort_return_swapped: }
978         { \sort_return_same: }
979     }
980     { \sort_return_same: }
981 }
982 }
983 }
984 }

```

(End definition for _zrefclever_sort_default:nn.)

_zrefclever_sort_default_same_type:nn

```

985 \cs_new_protected:Npn \_zrefclever_sort_default_same_type:nn #1#2
986 {
987     \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
988     { \zref@extractdefault {#1} { zc@enclcnt } { \c_empty_tl } }
989     \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
990     { \tl_reverse_items:V \l__zrefclever_label_enclcnt_a_tl }
991     \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
992     { \zref@extractdefault {#2} { zc@enclcnt } { \c_empty_tl } }
993     \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
994     { \tl_reverse_items:V \l__zrefclever_label_enclcnt_b_tl }
995     \tl_set:Nx \l__zrefclever_label_enclval_a_tl
996     { \zref@extractdefault {#1} { zc@enclval } { \c_empty_tl } }
997     \tl_set:Nx \l__zrefclever_label_enclval_a_tl
998     { \tl_reverse_items:V \l__zrefclever_label_enclval_a_tl }
999     \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1000     { \zref@extractdefault {#2} { zc@enclval } { \c_empty_tl } }
1001     \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1002     { \tl_reverse_items:V \l__zrefclever_label_enclval_b_tl }
1003
1004     \bool_set_false:N \l__zrefclever_sort_decided_bool
1005     % CHECK should I replace the tmp variables here?
1006     \tl_clear:N \l_tmpa_tl
1007     \tl_clear:N \l_tmpb_tl
1008     \bool_until_do:Nn \l__zrefclever_sort_decided_bool
1009     {
1010         \tl_set:Nx \l_tmpa_tl
1011         { \tl_head:N \l__zrefclever_label_enclcnt_a_tl }
1012         \tl_set:Nx \l_tmpb_tl
1013         { \tl_head:N \l__zrefclever_label_enclcnt_b_tl }
1014
1015         \bool_if:nTF
1016         {
1017             % Both are empty, meaning: neither labels have any (further)
1018             % ‘‘enclosing counters’’ (left).
1019             \tl_if_empty_p:V \l_tmpa_tl &&
1020             \tl_if_empty_p:V \l_tmpb_tl
1021         }
1022         {

```

```

1023 \exp_args:Nxx \tl_if_eq:nnTF
1024 { \zref@extractdefault {#1} { counter } { } }
1025 { \zref@extractdefault {#2} { counter } { } }
1026 {
1027   \bool_set_true:N \l__zrefclever_sort_decided_bool
1028   \int_compare:nNnTF
1029     { \zref@extractdefault {#1} { zc@cntval } {-1} }
1030     >
1031     { \zref@extractdefault {#2} { zc@cntval } {-1} }
1032     { \sort_return_swapped: }
1033     { \sort_return_same: }
1034 }
1035 {
1036   \msg_warning:nnnn { zref-clever }
1037   { counters-not-nested } {#1} {#2}
1038   \bool_set_true:N \l__zrefclever_sort_decided_bool
1039   \sort_return_same:
1040 }
1041 }
1042 {
1043   \bool_if:nTF
1044   {
1045     % 'a' is empty (and 'b' is not), meaning: 'b' is (possibly)
1046     % nested in 'a'.
1047     \tl_if_empty_p:V \l_tmpa_tl
1048   }
1049   {
1050     \tl_set:Nx \l_tmpa_tl
1051     { {\zref@extractdefault {#1} { counter } { } } }
1052     \exp_args:NNx \tl_if_in:NnTF
1053     \l__zrefclever_label_enclcnt_b_tl { \l_tmpa_tl }
1054     {
1055       \bool_set_true:N \l__zrefclever_sort_decided_bool
1056       \sort_return_same:
1057     }
1058     {
1059       \msg_warning:nnnn { zref-clever }
1060       { counters-not-nested } {#1} {#2}
1061       \bool_set_true:N \l__zrefclever_sort_decided_bool
1062       \sort_return_same:
1063     }
1064   }
1065 }
1066 \bool_if:nTF
1067 {
1068   % 'b' is empty (and 'a' is not), meaning: 'a' is
1069   % (possibly) nested in 'b'.
1070   \tl_if_empty_p:V \l_tmpb_tl
1071 }
1072 {
1073   \tl_set:Nx \l_tmpb_tl
1074   { {\zref@extractdefault {#2} { counter } { } } }
1075   \exp_args:NNx \tl_if_in:NnTF
1076   \l__zrefclever_label_enclcnt_a_tl { \l_tmpb_tl }

```

```

1077 {
1078   \bool_set_true:N \l__zrefclever_sort_decided_bool
1079   \sort_return_swapped:
1080 }
1081 {
1082   \msg_warning:nnnn { zref-clever }
1083   { counters-not-nested } {#1} {#2}
1084   \bool_set_true:N \l__zrefclever_sort_decided_bool
1085   \sort_return_same:
1086 }
1087 }
1088 {
1089   % Neither is empty, meaning: we can (possibly) compare the
1090   % values of the current enclosing counter in the loop, if
1091   % they are equal, we are still in the loop, if they are
1092   % not, a sorting decision can be made directly.
1093   \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
1094   {
1095     \int_compare:nNnTF
1096       { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1097       =
1098       { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1099       {
1100         \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1101           { \tl_tail:N \l__zrefclever_label_enclcnt_a_tl }
1102         \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1103           { \tl_tail:N \l__zrefclever_label_enclcnt_b_tl }
1104         \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1105           { \tl_tail:N \l__zrefclever_label_enclval_a_tl }
1106         \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1107           { \tl_tail:N \l__zrefclever_label_enclval_b_tl }
1108       }
1109       {
1110         \bool_set_true:N \l__zrefclever_sort_decided_bool
1111         \int_compare:nNnTF
1112           { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1113           >
1114           { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1115           { \sort_return_swapped: }
1116           { \sort_return_same: }
1117       }
1118     }
1119   {
1120     \msg_warning:nnnn { zref-clever }
1121     { counters-not-nested } {#1} {#2}
1122     \bool_set_true:N \l__zrefclever_sort_decided_bool
1123     \sort_return_same:
1124   }
1125 }
1126 }
1127 }
1128 }
1129 }

```

(End definition for __zrefclever_sort_default_same_type:nn.)

_zrefclever_sort_default_different_types:nn

```

1130 \cs_new_protected:Npn \__zrefclever_sort_default_different_types:nn #1#2
1131 {
1132   \int_zero:N \l__zrefclever_sort_prior_a_int
1133   \int_zero:N \l__zrefclever_sort_prior_b_int
1134   % \cs{l__zrefclever_typesort_seq} was stored in reverse sequence, and we compute
1135   % the sort priorities in the negative range, so that we can implicitly
1136   % rely on '0' being the 'last value'.
1137   \seq_map_indexed_inline:Nn \l__zrefclever_typesort_seq
1138   {
1139     \tl_if_eq:nnTF {##2} {{othertypes}}
1140     {
1141       \int_compare:nNnT { \l__zrefclever_sort_prior_a_int } = { 0 }
1142       { \int_set:Nn \l__zrefclever_sort_prior_a_int { - ##1 } }
1143       \int_compare:nNnT { \l__zrefclever_sort_prior_b_int } = { 0 }
1144       { \int_set:Nn \l__zrefclever_sort_prior_b_int { - ##1 } }
1145     }
1146     {
1147       \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##2}
1148       { \int_set:Nn \l__zrefclever_sort_prior_a_int { - ##1 } }
1149       {
1150         \tl_if_eq:NnT \l__zrefclever_label_type_b_tl {##2}
1151         { \int_set:Nn \l__zrefclever_sort_prior_b_int { - ##1 } }
1152       }
1153     }
1154   }
1155   \bool_if:nTF
1156   {
1157     \int_compare_p:nNn
1158     { \l__zrefclever_sort_prior_a_int } <
1159     { \l__zrefclever_sort_prior_b_int }
1160   }
1161   { \sort_return_same: }
1162   {
1163     \bool_if:nTF
1164     {
1165       \int_compare_p:nNn
1166       { \l__zrefclever_sort_prior_a_int } >
1167       { \l__zrefclever_sort_prior_b_int }
1168     }
1169     { \sort_return_swapped: }
1170     {
1171       % Sort priorities are equal for different types: the type that
1172       % occurs first in \meta{labels}, as given by the user, is kept (or
1173       % brought) forward.
1174       \seq_map_inline:Nn \l__zrefclever_label_types_seq
1175       {
1176         \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##1}
1177         { \seq_map_break:n { \sort_return_same: } }
1178         {
1179           \tl_if_eq:NnT \l__zrefclever_label_type_b_tl {##1}
1180           { \seq_map_break:n { \sort_return_swapped: } }
1181         }
1182       }
1183     }
1184   }

```



```

1183     }
1184   }
1185 }

```

(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}

1186 \cs_new_protected:Npn \__zrefclever_sort_page:nn #1#2
1187 {
1188   \int_compare:nNnTF
1189     { \zref@extractdefault {#1} { abspage } {-1} }
1190     >
1191     { \zref@extractdefault {#2} { abspage } {-1} }
1192     { \sort_return_swapped: }
1193     { \sort_return_same:   }
1194 }

```

(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.

```

1195 \bool_new:N \l__zrefclever_typeset_last_bool
1196 \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.

```

1197 \seq_new:N \l__zrefclever_typeset_labels_seq
1198 \tl_new:N \l__zrefclever_typeset_queue_prev_tl
1199 \tl_new:N \l__zrefclever_typeset_queue_curr_tl
1200 \tl_new:N \l__zrefclever_type_first_label_tl
1201 \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.

```

1202 \int_new:N \l__zrefclever_label_count_int
1203 \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.

```

1204 \int_new:N \l__zrefclever_range_count_int
1205 \int_new:N \l__zrefclever_range_same_count_int
1206 \tl_new:N \l__zrefclever_range_beg_label_tl
1207 \bool_new:N \l__zrefclever_next_maybe_range_bool
1208 \bool_new:N \l__zrefclever_next_is_same_bool
1209 \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.

```

1210 \tl_new:N \l__zrefclever_namefont_tl
1211 \tl_new:N \l__zrefclever_reffont_out_tl
1212 \tl_new:N \l__zrefclever_reffont_in_tl
1213
1214 \tl_new:N \l__zrefclever_namesep_tl
1215 \tl_new:N \l__zrefclever_rangesep_tl
1216 \tl_new:N \l__zrefclever_pairsep_tl
1217 \tl_new:N \l__zrefclever_listsep_tl
1218 \tl_new:N \l__zrefclever_lastsep_tl

```

```

1219 % 't' for 'type'
1220 \tl_new:N \l__zrefclever_tpairsep_tl
1221 \tl_new:N \l__zrefclever_tlistsep_tl
1222 \tl_new:N \l__zrefclever_tlastsep_tl
1223 \tl_new:N \l__zrefclever_notesep_tl
1224 \tl_new:N \l__zrefclever_refpre_out_tl
1225 \tl_new:N \l__zrefclever_refpos_out_tl
1226 \tl_new:N \l__zrefclever_refpre_in_tl
1227 \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:.

```

1228 \tl_new:N \l__zrefclever_type_name_tl
1229 \bool_new:N \l__zrefclever_name_in_link_bool
1230 \tl_new:N \l__zrefclever_name_format_tl
1231 \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.

```

1232 \cs_new_protected:Npn \__zrefclever_typeset_refs:
1233 {
1234   \seq_set_eq:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_zceref_labels_seq
1235   \tl_clear:N \l__zrefclever_typeset_queue_prev_tl
1236   \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1237   \tl_clear:N \l__zrefclever_type_first_label_tl
1238   \tl_clear:N \l__zrefclever_type_first_label_type_tl
1239   \tl_clear:N \l__zrefclever_range_beg_label_tl
1240   \int_zero:N \l__zrefclever_label_count_int
1241   \int_zero:N \l__zrefclever_type_count_int
1242   \int_zero:N \l__zrefclever_range_count_int
1243   \int_zero:N \l__zrefclever_range_same_count_int
1244
1245   % Get not-type-specific separators and refpre/pos options.
1246   \__zrefclever_get_option_with_transl:nN {tpairsep} \l__zrefclever_tpairsep_tl
1247   \__zrefclever_get_option_with_transl:nN {tlistsep} \l__zrefclever_tlistsep_tl
1248   \__zrefclever_get_option_with_transl:nN {tlastsep} \l__zrefclever_tlastsep_tl
1249   \__zrefclever_get_option_with_transl:nN {notesep} \l__zrefclever_notesep_tl
1250
1251   % Set the font option for this zceref call.
1252   \l__zrefclever_ref_typeset_font_tl
1253
1254   % Loop over the label list in sequence.
1255   \bool_set_false:N \l__zrefclever_typeset_last_bool
1256   \bool_until_do:Nn \l__zrefclever_typeset_last_bool
1257   {
1258     \seq_pop_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_a_tl
1259     \seq_if_empty:NTF \l__zrefclever_typeset_labels_seq
1260     {
1261       \tl_clear:N \l__zrefclever_label_b_tl

```

```

1262         \bool_set_true:N \l__zrefclever_typeset_last_bool
1263     }
1264     { \seq_get_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_b_tl }
1265
1266 \tl_if_eq:NnTF \l__zrefclever_ref_property_tl { page }
1267 {
1268     \tl_set:Nn \l__zrefclever_label_type_a_tl { page }
1269     \tl_set:Nn \l__zrefclever_label_type_b_tl { page }
1270 }
1271 {
1272     \tl_set:Nx \l__zrefclever_label_type_a_tl
1273     {
1274         \zref@extractdefault
1275         { \l__zrefclever_label_a_tl } { zc@type } { \c_empty_tl }
1276     }
1277     \tl_set:Nx \l__zrefclever_label_type_b_tl
1278     {
1279         \zref@extractdefault
1280         { \l__zrefclever_label_b_tl } { zc@type } { \c_empty_tl }
1281     }
1282 }
1283
1284 % First, we establish whether the ‘current label’ (i.e. ‘a’) is the
1285 % last one of its type. This can happen because the ‘next label’
1286 % (i.e. ‘b’) is of a different type (or different definition status),
1287 % or because we are at the end of the list.
1288 \bool_if:NTF \l__zrefclever_typeset_last_bool
1289 { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1290 {
1291     \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1292     {
1293         \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1294         { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1295         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1296     }
1297     {
1298         \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1299         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1300         {
1301             % Neither is undefined, we must check the types.
1302             \bool_if:nTF
1303             % Both empty: same ‘type’.
1304             {
1305                 \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1306                 \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1307             }
1308             { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1309             {
1310                 \bool_if:nTF
1311                 % Neither empty: compare types.
1312                 {
1313                     ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1314                     ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1315                 }

```

```

1316         {
1317             \tl_if_eq:NNTF
1318                 \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
1319                 { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1320                 { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1321         }
1322         % One empty, the other not: different ‘types’.
1323         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1324     }
1325 }
1326 }
1327 }
1328
1329 % Handle warnings in case of reference or type undefined.
1330 \zref@refused { \l__zrefclever_label_a_tl }
1331 \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1332 {}
1333 {
1334     \tl_if_empty:NT \l__zrefclever_label_type_a_tl
1335     {
1336         \msg_warning:nxx { zref-clever } { missing-type }
1337         { \l__zrefclever_label_a_tl }
1338     }
1339 }
1340
1341 % Get type-specific separators, refpre/pos and font options, once per
1342 % type.
1343 \int_compare:nNnT { \l__zrefclever_label_count_int } = { 0 }
1344 {
1345     \__zrefclever_get_option_plain:nN {namefont}          \l__zrefclever_namefont_tl
1346     \__zrefclever_get_option_plain:nN {reffont}          \l__zrefclever_reffont_out_tl
1347     \__zrefclever_get_option_plain:nN {reffont-in}       \l__zrefclever_reffont_in_tl
1348     \__zrefclever_get_option_with_transl:nN {namesep}    \l__zrefclever_namesep_tl
1349     \__zrefclever_get_option_with_transl:nN {rangesep}   \l__zrefclever_rangesep_tl
1350     \__zrefclever_get_option_with_transl:nN {pairsep}    \l__zrefclever_pairsep_tl
1351     \__zrefclever_get_option_with_transl:nN {listsep}    \l__zrefclever_listsep_tl
1352     \__zrefclever_get_option_with_transl:nN {lastsep}    \l__zrefclever_lastsep_tl
1353     \__zrefclever_get_option_with_transl:nN {refpre}     \l__zrefclever_refpre_out_tl
1354     \__zrefclever_get_option_with_transl:nN {refpos}     \l__zrefclever_refpos_out_tl
1355     \__zrefclever_get_option_with_transl:nN {refpre-in}  \l__zrefclever_refpre_in_tl
1356     \__zrefclever_get_option_with_transl:nN {refpos-in}  \l__zrefclever_refpos_in_tl
1357 }
1358
1359 % Here we send this to a couple of auxiliary functions for no other
1360 % reason than to keep this long function a little less unreadable.
1361 \bool_if:NTF \l__zrefclever_last_of_type_bool
1362 {
1363     % There exists no next label of the same type as the current.
1364     \__zrefclever_typeset_refs_aux_last_of_type:
1365 }
1366 {
1367     % There exists a next label of the same type as the current.
1368     \__zrefclever_typeset_refs_aux_not_last_of_type:
1369 }

```

```

1370     }
1371 }

```

(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.

```

1372 \cs_new_protected:Npn \_zrefclever_typeset_refs_aux_last_of_type:
1373 {
1374   % Process the current label to the current queue.
1375   \int_case:nnF { \l__zrefclever_label_count_int }
1376   {
1377     % It is the last label of its type, but also the first one, and that's
1378     % what matters here: just store it.
1379     { 0 }
1380     {
1381       \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1382       \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1383     }
1384
1385     % The last is the second: we have a pair (if not repeated).
1386     { 1 }
1387     {
1388       \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1389       {
1390         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1391         {
1392           \exp_not:V \l__zrefclever_pairsep_tl
1393           \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1394         }
1395       }
1396     }
1397   }
1398   % If neither the first, nor the second: we have the last label
1399   % on the current type list (if not repeated).
1400   {
1401     \int_case:nnF { \l__zrefclever_range_count_int }
1402     {
1403       % There was no range going on.
1404       {0}
1405       {
1406         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1407         {
1408           \exp_not:V \l__zrefclever_lastsep_tl
1409           \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1410         }
1411       }
1412       % Last in the range is also the second in it.
1413       {1}
1414       {
1415         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1416         {
1417           % We know 'range_beg_label' is not empty, since this is the
1418           % second element in the range, but the third or more in the
1419           % type list.

```

```

1420 \exp_not:V \l__zrefclever_listsep_tl
1421 \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1422 \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1423 {
1424   \exp_not:V \l__zrefclever_lastsep_tl
1425   \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1426 }
1427 }
1428 }
1429 }
1430 % Last in the range is third or more in it.
1431 {
1432   \int_case:nnF
1433   { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1434   {
1435     % Repetition, not a range.
1436     {0}
1437     {
1438       % If 'range_beg_label' is empty, it means it was also the
1439       % first of the type, and hence was already handled.
1440       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1441       {
1442         \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1443         {
1444           \exp_not:V \l__zrefclever_lastsep_tl
1445           \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1446         }
1447       }
1448     }
1449     % A 'range', but with no skipped value, treat as list.
1450     {1}
1451     {
1452       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1453       {
1454         % Ditto.
1455         \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1456         {
1457           \exp_not:V \l__zrefclever_listsep_tl
1458           \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1459         }
1460         \exp_not:V \l__zrefclever_lastsep_tl
1461         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1462       }
1463     }
1464   }
1465   {
1466     % An actual range.
1467     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1468     {
1469       % Ditto.
1470       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1471       {
1472         \exp_not:V \l__zrefclever_lastsep_tl
1473         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl

```

```

1474         }
1475         \exp_not:V \l__zrefclever_rangesep_tl
1476         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1477     }
1478 }
1479 }
1480 }
1481
1482 % Handle ‘‘range’’ option. The idea is simple: if the queue is not empty,
1483 % we replace it with the end of the range (or pair). We can still
1484 % retrieve the end of the range from \cs{l__zrefclever_label_a_tl} since we know to
1485 % be processing the last label of its type at this point.
1486 \bool_if:NT \l__zrefclever_typeset_range_bool
1487 {
1488     \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1489     {
1490         \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1491         { }
1492         {
1493             \msg_warning:nxx { zref-clever } { single-element-range }
1494             { \l__zrefclever_type_first_label_type_tl }
1495         }
1496     }
1497     {
1498         \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1499         \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1500         { }
1501         {
1502             \__zrefclever_labels_in_sequence:nn
1503             { \l__zrefclever_type_first_label_tl } { \l__zrefclever_label_a_tl }
1504         }
1505         \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1506         {
1507             \bool_if:NTF \l__zrefclever_next_maybe_range_bool
1508             { \exp_not:V \l__zrefclever_pairsep_tl }
1509             { \exp_not:V \l__zrefclever_rangesep_tl }
1510             \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1511         }
1512     }
1513 }
1514
1515 % Now that the type is finished, we can add the name and the first ref to
1516 % the queue. Or, if ‘‘typset’’ option is not ‘‘both’’, handle it here
1517 % too.
1518 \__zrefclever_type_name_setup:
1519 \bool_if:nTF
1520 { \l__zrefclever_typeset_ref_bool && \l__zrefclever_typeset_name_bool }
1521 {
1522     \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1523     { \__zrefclever_get_ref_first: }
1524 }
1525 {
1526     \bool_if:nTF
1527     { \l__zrefclever_typeset_ref_bool }

```



```

1528 {
1529   \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1530   { \__zrefclever_get_ref:V \l__zrefclever_type_first_label_tl }
1531 }
1532 {
1533   \bool_if:nTF
1534     { \l__zrefclever_typeset_name_bool }
1535     {
1536       \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1537       {
1538         \bool_if:NTF \l__zrefclever_name_in_link_bool
1539         {
1540           \exp_not:N \group_begin:
1541           \exp_not:V \l__zrefclever_namefont_tl
1542           % It's two '@s', but escaped for DocStrip.
1543           \exp_not:N \hyper@@link
1544           {
1545             \zref@ifrefcontainsprop
1546               { \l__zrefclever_type_first_label_tl } { urluse }
1547             {
1548               \zref@extractdefault
1549                 { \l__zrefclever_type_first_label_tl }
1550                 { urluse } {}
1551             }
1552             {
1553               \zref@extractdefault
1554                 { \l__zrefclever_type_first_label_tl }
1555                 { url } {}
1556             }
1557           }
1558           {
1559             \zref@extractdefault
1560               { \l__zrefclever_type_first_label_tl } { anchor } {}
1561           }
1562           { \exp_not:V \l__zrefclever_type_name_tl }
1563           \exp_not:N \group_end:
1564         }
1565         {
1566           \exp_not:N \group_begin:
1567           \exp_not:V \l__zrefclever_namefont_tl
1568           \exp_not:V \l__zrefclever_type_name_tl
1569           \exp_not:N \group_end:
1570         }
1571       }
1572     }
1573   {
1574     % This case would correspond to "typeset=none" but should not
1575     % happen, given the options are set up to typeset at least one
1576     % of "ref" or "name", but a sensible fallback, equal to the
1577     % behavior of "both".
1578     \tl_put_left:Nx
1579       \l__zrefclever_typeset_queue_curr_tl { \__zrefclever_get_ref_first: }
1580   }
1581 }

```

```

1582     }
1583
1584     % Typeset the previous type, if there is one.
1585     \int_compare:nNnT { \l__zrefclever_type_count_int } > { 0 }
1586     {
1587         \int_compare:nNnT { \l__zrefclever_type_count_int } > { 1 }
1588         { \l__zrefclever_tlistsep_tl }
1589         \l__zrefclever_typeset_queue_prev_tl
1590     }
1591
1592     % Wrap up loop, or prepare for next iteration.
1593     \bool_if:NTF \l__zrefclever_typeset_last_bool
1594     {
1595         % We are finishing, typeset the current queue.
1596         \int_case:nnF { \l__zrefclever_type_count_int }
1597         {
1598             % Single type.
1599             { 0 }
1600             { \l__zrefclever_typeset_queue_curr_tl }
1601             % Pair of types.
1602             { 1 }
1603             {
1604                 \l__zrefclever_tpairsep_tl
1605                 \l__zrefclever_typeset_queue_curr_tl
1606             }
1607         }
1608         {
1609             % Last in list of types.
1610             \l__zrefclever_tlastsep_tl
1611             \l__zrefclever_typeset_queue_curr_tl
1612         }
1613     }
1614     {
1615         % There are further labels, set variables for next iteration.
1616         \tl_set_eq:NN
1617         \l__zrefclever_typeset_queue_prev_tl \l__zrefclever_typeset_queue_curr_tl
1618         \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1619         \tl_clear:N \l__zrefclever_type_first_label_tl
1620         \tl_clear:N \l__zrefclever_type_first_label_type_tl
1621         \tl_clear:N \l__zrefclever_range_beg_label_tl
1622         \int_zero:N \l__zrefclever_label_count_int
1623         \int_incr:N \l__zrefclever_type_count_int
1624         \int_zero:N \l__zrefclever_range_count_int
1625         \int_zero:N \l__zrefclever_range_same_count_int
1626     }
1627 }

```

(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.

```

1628 \cs_new_protected:Npn \__zrefclever_typeset_refs_aux_not_last_of_type:
1629 {
1630     % Signal if next label may form a range with the current one (of
1631     % course, only considered if compression is enabled in the first

```

```

1632 % place).
1633 \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1634 \bool_set_false:N \l__zrefclever_next_is_same_bool
1635 \bool_lazy_and:nnT
1636 { \l__zrefclever_typeset_compress_bool }
1637 % Currently no-op, but kept as ‘handle’ to inhibit compression of
1638 % individual labels.
1639 { ! \l__zrefclever_range_inhibit_next_bool }
1640 {
1641   \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1642   { }
1643   {
1644     \__zrefclever_labels_in_sequence:nn
1645     { \l__zrefclever_label_a_tl } { \l__zrefclever_label_b_tl }
1646   }
1647 }
1648
1649 % Process the current label to the current queue.
1650 \int_compare:nNnTF { \l__zrefclever_label_count_int } = { 0 }
1651 {
1652   % Current label is the first of its type (also not the last, but it
1653   % doesn’t matter here): just store the label.
1654   \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1655   \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1656
1657   % If the next label may be part of a range, we set ‘range_beg_label’
1658   % to ‘empty’ (we deal with it as the ‘first’, and must do it
1659   % there, to handle hyperlinking), but also step the range counters.
1660   \bool_if:NT \l__zrefclever_next_maybe_range_bool
1661   {
1662     \tl_clear:N \l__zrefclever_range_beg_label_tl
1663     \int_incr:N \l__zrefclever_range_count_int
1664     \bool_if:NT \l__zrefclever_next_is_same_bool
1665     { \int_incr:N \l__zrefclever_range_same_count_int }
1666   }
1667 }
1668 {
1669   % Current label is neither the first (nor the last) of its
1670   % type.
1671   \bool_if:NNTF \l__zrefclever_next_maybe_range_bool
1672   {
1673     % Starting, or continuing a range.
1674     \int_compare:nNnTF
1675     { \l__zrefclever_range_count_int } = {0}
1676     {
1677       % There was no range going, we are starting one.
1678       \tl_set:NV \l__zrefclever_range_beg_label_tl \l__zrefclever_label_a_tl
1679       \int_incr:N \l__zrefclever_range_count_int
1680       \bool_if:NT \l__zrefclever_next_is_same_bool
1681       { \int_incr:N \l__zrefclever_range_same_count_int }
1682     }
1683     {
1684       % Second or more in the range, but not the last.
1685       \int_incr:N \l__zrefclever_range_count_int

```

```

1686         \bool_if:NT \l__zrefclever_next_is_same_bool
1687         { \int_incr:N \l__zrefclever_range_same_count_int }
1688     }
1689 }
1690 {
1691     % Next element is not in sequence, meaning: there was no range, or
1692     % we are closing one.
1693     \int_case:nnF { \l__zrefclever_range_count_int }
1694     {
1695         % There was no range going on.
1696         {0}
1697         {
1698             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1699             {
1700                 \exp_not:V \l__zrefclever_listsep_tl
1701                 \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1702             }
1703         }
1704         % Last is second in the range: if 'range_same_count' is also
1705         % '1', it's a repetition (drop it), otherwise, it's a "pair
1706         % within a list", treat as list.
1707         {1}
1708         {
1709             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1710             {
1711                 \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1712                 {
1713                     \exp_not:V \l__zrefclever_listsep_tl
1714                     \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1715                 }
1716                 \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1717                 {
1718                     \exp_not:V \l__zrefclever_listsep_tl
1719                     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1720                 }
1721             }
1722         }
1723     }
1724 }
1725 {
1726     % Last is third or more in the range: if 'range_count' and
1727     % 'range_same_count' are the same, its a repetition (drop it),
1728     % if they differ by '1', its a list, if they differ by more,
1729     % it is a real range.
1730     \int_case:nnF
1731     { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1732     {
1733         {0}
1734         {
1735             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1736             {
1737                 \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1738                 {
1739                     \exp_not:V \l__zrefclever_listsep_tl
1740                     \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1741                 }
1742             }
1743         }
1744     }
1745 }

```

```

1740     }
1741   }
1742 }
1743 {1}
1744 {
1745   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1746   {
1747     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1748     {
1749       \exp_not:V \l__zrefclever_listsep_tl
1750       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1751     }
1752     \exp_not:V \l__zrefclever_listsep_tl
1753     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1754   }
1755 }
1756 }
1757 {
1758   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1759   {
1760     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1761     {
1762       \exp_not:V \l__zrefclever_listsep_tl
1763       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1764     }
1765     \exp_not:V \l__zrefclever_rangesep_tl
1766     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1767   }
1768 }
1769 }
1770 % Reset counters.
1771 \int_zero:N \l__zrefclever_range_count_int
1772 \int_zero:N \l__zrefclever_range_same_count_int
1773 }
1774 }
1775 % Step label counter for next iteration.
1776 \int_incr:N \l__zrefclever_label_count_int
1777 }

```

(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.

```

1778 \cs_new:Npn \__zrefclever_get_ref:n #1
1779 {
1780   \zref@ifrefcontainsprop {#1} { \l__zrefclever_ref_property_tl }
1781   {
1782     \bool_if:nTF

```

```

1783 { \l__zrefclever_use_hyperref_bool && ! \l__zrefclever_link_star_bool }
1784 {
1785   \exp_not:N \group_begin:
1786   \exp_not:V \l__zrefclever_reffont_out_tl
1787   \exp_not:V \l__zrefclever_refpre_out_tl
1788   \exp_not:N \group_begin:
1789   \exp_not:V \l__zrefclever_reffont_in_tl
1790   % It's two '@s', but escaped for DocStrip.
1791   \exp_not:N \hyper@@link
1792   {
1793     \zref@ifrefcontainsprop {#1} { urluse }
1794     { \zref@extractdefault {#1} { urluse } {} }
1795     { \zref@extractdefault {#1} { url } {} }
1796   }
1797   { \zref@extractdefault {#1} { anchor } {} }
1798   {
1799     \exp_not:V \l__zrefclever_refpre_in_tl
1800     \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1801     \exp_not:V \l__zrefclever_refpos_in_tl
1802   }
1803   \exp_not:N \group_end:
1804   \exp_not:V \l__zrefclever_refpos_out_tl
1805   \exp_not:N \group_end:
1806 }
1807 {
1808   \exp_not:N \group_begin:
1809   \exp_not:V \l__zrefclever_reffont_out_tl
1810   \exp_not:V \l__zrefclever_refpre_out_tl
1811   \exp_not:N \group_begin:
1812   \exp_not:V \l__zrefclever_reffont_in_tl
1813   \exp_not:V \l__zrefclever_refpre_in_tl
1814   \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1815   \exp_not:V \l__zrefclever_refpos_in_tl
1816   \exp_not:N \group_end:
1817   \exp_not:V \l__zrefclever_refpos_out_tl
1818   \exp_not:N \group_end:
1819 }
1820 }
1821 { \exp_not:N \zref@default }
1822 }
1823 \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.

```

1824 \cs_new_protected:Npn \__zrefclever_type_name_setup:
1825 {
1826   \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1827   { \tl_clear:N \l__zrefclever_type_name_tl }
1828   {
1829     \tl_if_empty:nTF \l__zrefclever_type_first_label_type_tl
1830     { \tl_clear:N \l__zrefclever_type_name_tl }
1831     {

```

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

```

1832     \bool_lazy_or:nnTF
1833     { \l__zrefclever_capitalize_bool }
1834     {
1835         \l__zrefclever_capitalize_first_bool &&
1836         \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1837     }
1838     { \tl_set:Nn \l__zrefclever_name_format_tl {Name} }
1839     { \tl_set:Nn \l__zrefclever_name_format_tl {name} }
1840     % If the queue is empty, we have a singular, otherwise, plural.
1841     \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1842     { \tl_put_right:Nn \l__zrefclever_name_format_tl { -sg } }
1843     { \tl_put_right:Nn \l__zrefclever_name_format_tl { -pl } }
1844     \bool_lazy_and:nnTF
1845     { \l__zrefclever_abbrev_bool }
1846     {
1847         ! \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 } ||
1848         ! \l__zrefclever_noabbrev_first_bool
1849     }
1850     {
1851         \tl_set:NV \l__zrefclever_name_format_fallback_tl \l__zrefclever_name_format
1852         \tl_put_right:Nn \l__zrefclever_name_format_tl { -ab }
1853     }
1854     { \tl_clear:N \l__zrefclever_name_format_fallback_tl }
1855
1856     \tl_if_empty:NTF \l__zrefclever_name_format_fallback_tl
1857     {
1858         \prop_get:cVNF
1859         { l__zrefclever_type \l__zrefclever_type_first_label_type_tl _options_pro
1860         \l__zrefclever_name_format_tl
1861         \l__zrefclever_type_name_tl
1862         {
1863             \__zrefclever_if_transl:xxTF
1864             { \l__zrefclever_ref_language_tl }
1865             {
1866                 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1867                 \l__zrefclever_name_format_tl
1868             }
1869             {
1870                 \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1871                 { \l__zrefclever_ref_language_tl }
1872                 {
1873                     zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1874                     \l__zrefclever_name_format_tl
1875                 }
1876             }
1877         }
1878         \tl_clear:N \l__zrefclever_type_name_tl
1879         \msg_warning:nxx { zref-clever } { missing-name }
1880         { \l__zrefclever_type_first_label_type_tl }
1881     }
1882 }
1883 }
1884 {

```

```

1885 \prop_get:cVNF
1886 { l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1887 \l__zrefclever_name_format_tl
1888 \l__zrefclever_type_name_tl
1889 {
1890 \prop_get:cVNF
1891 { l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options
1892 \l__zrefclever_name_format_fallback_tl
1893 \l__zrefclever_type_name_tl
1894 {
1895 \__zrefclever_if_transl:xxTF
1896 { \l__zrefclever_ref_language_tl }
1897 {
1898 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1899 \l__zrefclever_name_format_tl
1900 }
1901 {
1902 \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1903 { \l__zrefclever_ref_language_tl }
1904 {
1905 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1906 \l__zrefclever_name_format_tl
1907 }
1908 }
1909 {
1910 \__zrefclever_if_transl:xxTF
1911 { \l__zrefclever_ref_language_tl }
1912 {
1913 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1914 \l__zrefclever_name_format_fallback_tl
1915 }
1916 {
1917 \__zrefclever_get_transl:nxx { \l__zrefclever_type_name_tl }
1918 { \l__zrefclever_ref_language_tl }
1919 {
1920 zrefclever-type- \l__zrefclever_type_first_label_type_tl
1921 \l__zrefclever_name_format_fallback_tl
1922 }
1923 }
1924 {
1925 \tl_clear:N \l__zrefclever_type_name_tl
1926 \msg_warning:nxx { zref-clever } { missing-name }
1927 { \l__zrefclever_type_first_label_type_tl }
1928 }
1929 }
1930 }
1931 }
1932 }
1933 }
1934 }
Signal whether the type name is to be included in the hyperlink or not.
1935 \bool_lazy_any:nTF
1936 {
1937 { ! \l__zrefclever_use_hyperref_bool }

```



```

1938 { \l__zrefclever_link_star_bool }
1939 { \tl_if_empty_p:N \l__zrefclever_type_name_tl }
1940 { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { false } }
1941 }
1942 { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1943 {
1944   \bool_lazy_any:nTF
1945   {
1946     { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { true } }
1947     {
1948       \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { tsingle } &&
1949       \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl
1950     }
1951     {
1952       \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { single } &&
1953       \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl &&
1954       \l__zrefclever_typeset_last_bool &&
1955       \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1956     }
1957   }
1958   { \bool_set_true:N \l__zrefclever_name_in_link_bool }
1959   { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1960 }
1961 }

```

(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.

```

1962 \cs_new:Npn \__zrefclever_get_ref_first:
1963 {
1964   \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1965   { \exp_not:N \zref@default }
1966   {
1967     \bool_if:NTF \l__zrefclever_name_in_link_bool
1968     {
1969       \zref@ifrefcontainsprop
1970       { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
1971       {
1972         % It's two '@s', but escaped for DocStrip.
1973         \exp_not:N \hyper@@link
1974         {
1975           \zref@ifrefcontainsprop
1976           { \l__zrefclever_type_first_label_tl } { urluse }
1977           {
1978             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
1979             { urluse } {}
1980           }
1981           {
1982             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
1983             { url } {}
1984           }
1985         }
1986       }
1987     }
1988   }

```

```

1986 {
1987   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
1988   { anchor } {}
1989 }
1990 {
1991   \exp_not:N \group_begin:
1992   \exp_not:V \l__zrefclever_namefont_tl
1993   \exp_not:V \l__zrefclever_type_name_tl
1994   \exp_not:N \group_end:
1995   \exp_not:V \l__zrefclever_namesep_tl
1996   \exp_not:N \group_begin:
1997   \exp_not:V \l__zrefclever_reffont_out_tl
1998   \exp_not:V \l__zrefclever_refpre_out_tl
1999   \exp_not:N \group_begin:
2000   \exp_not:V \l__zrefclever_reffont_in_tl
2001   \exp_not:V \l__zrefclever_refpre_in_tl
2002   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2003   { \l__zrefclever_ref_property_tl } {}
2004   \exp_not:V \l__zrefclever_refpos_in_tl
2005   \exp_not:N \group_end:
2006   % hyperlink makes it's own group, we'd like to close the
2007   % 'refpre-out' group after 'refpos-out', but... we close
2008   % it here, and give the trailing 'refpos-out' its own
2009   % group. This will result that formatting given to
2010   % 'refpre-out' will not reach 'refpos-out', but I see no
2011   % alternative, and this has to be handled specially.
2012   \exp_not:N \group_end:
2013 }
2014 \exp_not:N \group_begin:
2015 % Ditto: special treatment.
2016 \exp_not:V \l__zrefclever_reffont_out_tl
2017 \exp_not:V \l__zrefclever_refpos_out_tl
2018 \exp_not:N \group_end:
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 \zref@default
2027 }
2028 }
2029 {
2030   \tl_if_empty:NTF \l__zrefclever_type_name_tl
2031   {
2032     \exp_not:N \zref@default
2033     \exp_not:V \l__zrefclever_namesep_tl
2034   }
2035   {
2036     \exp_not:N \group_begin:
2037     \exp_not:V \l__zrefclever_namefont_tl
2038     \exp_not:V \l__zrefclever_type_name_tl
2039     \exp_not:N \group_end:

```

```

2040         \exp_not:V \l__zrefclever_namesep_tl
2041     }
2042 \zref@ifrefcontainsprop
2043 { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
2044 {
2045     \bool_if:nTF
2046     {
2047         \l__zrefclever_use_hyperref_bool &&
2048         ! \l__zrefclever_link_star_bool
2049     }
2050     {
2051         \exp_not:N \group_begin:
2052         \exp_not:V \l__zrefclever_reffont_out_tl
2053         \exp_not:V \l__zrefclever_refpre_out_tl
2054         \exp_not:N \group_begin:
2055         \exp_not:V \l__zrefclever_reffont_in_tl
2056         % It's two '@s', but escaped for DocStrip.
2057         \exp_not:N \hyper@@link
2058         {
2059             \zref@ifrefcontainsprop
2060             { \l__zrefclever_type_first_label_tl } { urluse }
2061             {
2062                 \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2063                 { urluse } {}
2064             }
2065             {
2066                 \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2067                 { url } {}
2068             }
2069         }
2070         {
2071             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2072             { anchor } {}
2073         }
2074         {
2075             \exp_not:V \l__zrefclever_refpre_in_tl
2076             \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2077             { \l__zrefclever_ref_property_tl } {}
2078             \exp_not:V \l__zrefclever_refpos_in_tl
2079         }
2080         \exp_not:N \group_end:
2081         \exp_not:V \l__zrefclever_refpos_out_tl
2082         \exp_not:N \group_end:
2083     }
2084     {
2085         \exp_not:N \group_begin:
2086         \exp_not:V \l__zrefclever_reffont_out_tl
2087         \exp_not:V \l__zrefclever_refpre_out_tl
2088         \exp_not:N \group_begin:
2089         \exp_not:V \l__zrefclever_reffont_in_tl
2090         \exp_not:V \l__zrefclever_refpre_in_tl
2091         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2092         { \l__zrefclever_ref_property_tl } {}
2093         \exp_not:V \l__zrefclever_refpos_in_tl

```

```

2094         \exp_not:N \group_end:
2095         \exp_not:V \l__zrefclever_refpos_out_tl
2096         \exp_not:N \group_end:
2097     }
2098 }
2099 { \exp_not:N \zref@default }
2100 }
2101 }
2102 }

```

(End definition for __zrefclever_get_ref_first:.)

__zrefclever_get_option_with_transl:nN

```

2103 % \Arg{option} \Arg{var to store result}
2104 \cs_new_protected:Npn \__zrefclever_get_option_with_transl:nN #1#2
2105 {
2106     % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.
2107     \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2108     {
2109         % If not found, try the type specific options.
2110         \bool_lazy_all:nTF
2111         {
2112             { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2113             {
2114                 \prop_if_exist_p:c
2115                 { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2116             }
2117             {
2118                 \prop_if_in_p:cn
2119                 { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1}
2120             }
2121         }
2122         {
2123             \prop_get:cnN
2124             { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2125         }
2126         {
2127             % If not found, try the type specific translations.
2128             \__zrefclever_if_transl:xxTF
2129             { \l__zrefclever_ref_language_tl }
2130             { zrefclever-type- \l__zrefclever_label_type_a_tl - #1 }
2131             {
2132                 \__zrefclever_get_transl:nxx {#2}
2133                 { \l__zrefclever_ref_language_tl }
2134                 { zrefclever-type- \l__zrefclever_label_type_a_tl - #1 }
2135             }
2136             {
2137                 % If not found, try general translations. We are not
2138                 % controlling for their existence, but we must make sure all
2139                 % options being retrieved with
2140                 % \cs{l__zrefclever_get_option_with_transl:nN} have their values set for
2141                 % 'English' and 'fallback'.
2142                 \__zrefclever_get_transl:nxx {#2}
2143                 { \l__zrefclever_ref_language_tl }

```

```

2144         { zrefclever-default- #1 }
2145     }
2146 }
2147 }
2148 }

```

(End definition for _zrefclever_get_option_with_transl:nN.)

_zrefclever_get_option_plain:nN

```

2149 \cs_new_protected:Npn \_zrefclever_get_option_plain:nN #1#2
2150 {
2151     % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.
2152     \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2153     {
2154         % If not found, try the type specific options.
2155         \bool_lazy_and:nnTF
2156         { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2157         {
2158             \prop_if_exist_p:c
2159             { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2160         }
2161         {
2162             \prop_get:cnNF
2163             { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2164             { \tl_clear:N #2 }
2165         }
2166         { \tl_clear:N #2 }
2167     }
2168 }

```

(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”.

```

2169 \cs_new_protected:Npn \_zrefclever_labels_in_sequence:nn #1#2
2170 {
2171     \tl_if_eq:NnTF \l__zrefclever_ref_property_tl { page }
2172     {
2173         \exp_args:Nxx \tl_if_eq:nnT
2174         { \zref@extractdefault {#1} { zc@pgfmt } { } }
2175         { \zref@extractdefault {#2} { zc@pgfmt } { } }
2176         {
2177             \int_compare:nNnTF
2178             { \zref@extractdefault {#1} { zc@pgval } {-2} + 1 }
2179             =
2180             { \zref@extractdefault {#2} { zc@pgval } {-1} }
2181             { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2182             {
2183                 \int_compare:nNnT
2184                 { \zref@extractdefault {#1} { zc@pgval } {-1} }
2185                 =
2186                 { \zref@extractdefault {#2} { zc@pgval } {-1} }
2187                 {

```

```

2188         \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2189         \bool_set_true:N \l__zrefclever_next_is_same_bool
2190     }
2191 }
2192 }
2193 }
2194 {
2195     \exp_args:Nxx \tl_if_eq:nnT
2196     { \zref@extractdefault {#1} { counter } { } }
2197     { \zref@extractdefault {#2} { counter } { } }
2198     {
2199         \exp_args:Nxx \tl_if_eq:nnT
2200         { \zref@extractdefault {#1} { zc@enclval } { } }
2201         { \zref@extractdefault {#2} { zc@enclval } { } }
2202         {
2203             \int_compare:nNnTF
2204             { \zref@extractdefault {#1} { zc@cntval } {-2} + 1 }
2205             =
2206             { \zref@extractdefault {#2} { zc@cntval } {-1} }
2207             { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2208             {
2209                 \int_compare:nNnT
2210                 { \zref@extractdefault {#1} { zc@cntval } {-1} }
2211                 =
2212                 { \zref@extractdefault {#2} { zc@cntval } {-1} }
2213                 {
2214                     \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2215                     \bool_set_true:N \l__zrefclever_next_is_same_bool
2216                 }
2217             }
2218         }
2219     }
2220 }
2221 }

```

(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’.

```

2222 \__zrefclever_declare_default_transl:nnn { fallback } { namesep } { \nobreakspace }
2223 \__zrefclever_declare_default_transl:nnn { fallback } { pairsep } { {,~} }
2224 \__zrefclever_declare_default_transl:nnn { fallback } { listsep } { {,~} }
2225 \__zrefclever_declare_default_transl:nnn { fallback } { lastsep } { {,~} }
2226 \__zrefclever_declare_default_transl:nnn { fallback } { tpairsep } { {,~} }
2227 \__zrefclever_declare_default_transl:nnn { fallback } { tlistsep } { {,~} }
2228 \__zrefclever_declare_default_transl:nnn { fallback } { tlastsep } { {,~} }
2229 \__zrefclever_declare_default_transl:nnn { fallback } { notesep } { {~} }
2230 \__zrefclever_declare_default_transl:nnn { fallback } { rangesep } { { \textendash } }
2231 \__zrefclever_declare_default_transl:nnn { fallback } { refpre } { {} }
2232 \__zrefclever_declare_default_transl:nnn { fallback } { refpos } { {} }
2233 \__zrefclever_declare_default_transl:nnn { fallback } { refpre-in } { {} }
2234 \__zrefclever_declare_default_transl:nnn { fallback } { refpos-in } { {} }
2235 </package>
2236 <*lang-english>

```

English

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

```

2237 \ProvideDictionaryFor{English}{zref-clever}
2238
2239 \zcDicDefaultTransl{namesep}{ \nobreakspace }
2240 \zcDicDefaultTransl{pairsep}{ {~and\nobreakspace } }
2241 \zcDicDefaultTransl{listsep}{ {,~} }
2242 \zcDicDefaultTransl{lastsep}{ {~and\nobreakspace } }
2243 \zcDicDefaultTransl{tpairsep}{ {~and\nobreakspace } }
2244 \zcDicDefaultTransl{tlistsep}{ {,~} }
2245 \zcDicDefaultTransl{tlastsep}{ {,~and\nobreakspace } }

```

2246 \zcDicDefaultTransl{notesep}{~}
2247 \zcDicDefaultTransl{rangesep}{~to\nobreakspace}
2248 \zcDicDefaultTransl{refpre}{}
2249 \zcDicDefaultTransl{refpos}{}
2250 \zcDicDefaultTransl{refpre-in}{}
2251 \zcDicDefaultTransl{refpos-in}{}
2252
2253 \zcDicTypeTransl{part}{Name-sg}{Part}
2254 \zcDicTypeTransl{part}{name-sg}{part}
2255 \zcDicTypeTransl{part}{Name-pl}{Parts}
2256 \zcDicTypeTransl{part}{name-pl}{parts}
2257
2258 \zcDicTypeTransl{chapter}{Name-sg}{Chapter}
2259 \zcDicTypeTransl{chapter}{name-sg}{chapter}
2260 \zcDicTypeTransl{chapter}{Name-pl}{Chapters}
2261 \zcDicTypeTransl{chapter}{name-pl}{chapters}
2262
2263 \zcDicTypeTransl{section}{Name-sg}{Section}
2264 \zcDicTypeTransl{section}{name-sg}{section}
2265 \zcDicTypeTransl{section}{Name-pl}{Sections}
2266 \zcDicTypeTransl{section}{name-pl}{sections}
2267
2268 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraph}
2269 \zcDicTypeTransl{paragraph}{name-sg}{paragraph}
2270 \zcDicTypeTransl{paragraph}{Name-pl}{Paragraphs}
2271 \zcDicTypeTransl{paragraph}{name-pl}{paragraphs}
2272 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}
2273 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}
2274 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}
2275 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}
2276
2277 \zcDicTypeTransl{appendix}{Name-sg}{Appendix}
2278 \zcDicTypeTransl{appendix}{name-sg}{appendix}
2279 \zcDicTypeTransl{appendix}{Name-pl}{Appendices}
2280 \zcDicTypeTransl{appendix}{name-pl}{appendices}
2281
2282 \zcDicTypeTransl{page}{Name-sg}{Page}
2283 \zcDicTypeTransl{page}{name-sg}{page}
2284 \zcDicTypeTransl{page}{Name-pl}{Pages}
2285 \zcDicTypeTransl{page}{name-pl}{pages}
2286 \zcDicTypeTransl{page}{name-sg-ab}{p.}
2287 \zcDicTypeTransl{page}{name-pl-ab}{pp.}
2288
2289 \zcDicTypeTransl{line}{Name-sg}{Line}
2290 \zcDicTypeTransl{line}{name-sg}{line}
2291 \zcDicTypeTransl{line}{Name-pl}{Lines}
2292 \zcDicTypeTransl{line}{name-pl}{lines}
2293
2294 \zcDicTypeTransl{figure}{Name-sg}{Figure}
2295 \zcDicTypeTransl{figure}{name-sg}{figure}
2296 \zcDicTypeTransl{figure}{Name-pl}{Figures}
2297 \zcDicTypeTransl{figure}{name-pl}{figures}
2298 \zcDicTypeTransl{figure}{Name-sg-ab}{Fig.}
2299 \zcDicTypeTransl{figure}{name-sg-ab}{fig.}

2300 \zcDicTypeTransl{figure}{Name-pl-ab}{Figs.}
 2301 \zcDicTypeTransl{figure}{name-pl-ab}{figs.}
 2302
 2303 \zcDicTypeTransl{table}{Name-sg}{Table}
 2304 \zcDicTypeTransl{table}{name-sg}{table}
 2305 \zcDicTypeTransl{table}{Name-pl}{Tables}
 2306 \zcDicTypeTransl{table}{name-pl}{tables}
 2307
 2308 \zcDicTypeTransl{item}{Name-sg}{Item}
 2309 \zcDicTypeTransl{item}{name-sg}{item}
 2310 \zcDicTypeTransl{item}{Name-pl}{Items}
 2311 \zcDicTypeTransl{item}{name-pl}{items}
 2312
 2313 \zcDicTypeTransl{footnote}{Name-sg}{Footnote}
 2314 \zcDicTypeTransl{footnote}{name-sg}{footnote}
 2315 \zcDicTypeTransl{footnote}{Name-pl}{Footnotes}
 2316 \zcDicTypeTransl{footnote}{name-pl}{footnotes}
 2317
 2318 \zcDicTypeTransl{note}{Name-sg}{Note}
 2319 \zcDicTypeTransl{note}{name-sg}{note}
 2320 \zcDicTypeTransl{note}{Name-pl}{Notes}
 2321 \zcDicTypeTransl{note}{name-pl}{notes}
 2322
 2323 \zcDicTypeTransl{equation}{Name-sg}{Equation}
 2324 \zcDicTypeTransl{equation}{name-sg}{equation}
 2325 \zcDicTypeTransl{equation}{Name-pl}{Equations}
 2326 \zcDicTypeTransl{equation}{name-pl}{equations}
 2327 \zcDicTypeTransl{equation}{Name-sg-ab}{Eq.}
 2328 \zcDicTypeTransl{equation}{name-sg-ab}{eq.}
 2329 \zcDicTypeTransl{equation}{Name-pl-ab}{Eqs.}
 2330 \zcDicTypeTransl{equation}{name-pl-ab}{eqs.}
 2331 \zcDicTypeTransl{equation}{refpre-in}{(}
 2332 \zcDicTypeTransl{equation}{refpos-in}{)}
 2333
 2334 \zcDicTypeTransl{theorem}{Name-sg}{Theorem}
 2335 \zcDicTypeTransl{theorem}{name-sg}{theorem}
 2336 \zcDicTypeTransl{theorem}{Name-pl}{Theorems}
 2337 \zcDicTypeTransl{theorem}{name-pl}{theorems}
 2338
 2339 \zcDicTypeTransl{lemma}{Name-sg}{Lemma}
 2340 \zcDicTypeTransl{lemma}{name-sg}{lemma}
 2341 \zcDicTypeTransl{lemma}{Name-pl}{Lemmas}
 2342 \zcDicTypeTransl{lemma}{name-pl}{lemmas}
 2343
 2344 \zcDicTypeTransl{corollary}{Name-sg}{Corollary}
 2345 \zcDicTypeTransl{corollary}{name-sg}{corollary}
 2346 \zcDicTypeTransl{corollary}{Name-pl}{Corollaries}
 2347 \zcDicTypeTransl{corollary}{name-pl}{corollaries}
 2348
 2349 \zcDicTypeTransl{proposition}{Name-sg}{Proposition}
 2350 \zcDicTypeTransl{proposition}{name-sg}{proposition}
 2351 \zcDicTypeTransl{proposition}{Name-pl}{Propositions}
 2352 \zcDicTypeTransl{proposition}{name-pl}{propositions}
 2353

```

2354 \zcDicTypeTransl{definition}{Name-sg}{Definition}
2355 \zcDicTypeTransl{definition}{name-sg}{definition}
2356 \zcDicTypeTransl{definition}{Name-pl}{Definitions}
2357 \zcDicTypeTransl{definition}{name-pl}{definitions}
2358
2359 \zcDicTypeTransl{proof}{Name-sg}{Proof}
2360 \zcDicTypeTransl{proof}{name-sg}{proof}
2361 \zcDicTypeTransl{proof}{Name-pl}{Proofs}
2362 \zcDicTypeTransl{proof}{name-pl}{proofs}
2363
2364 \zcDicTypeTransl{result}{Name-sg}{Result}
2365 \zcDicTypeTransl{result}{name-sg}{result}
2366 \zcDicTypeTransl{result}{Name-pl}{Results}
2367 \zcDicTypeTransl{result}{name-pl}{results}
2368
2369 \zcDicTypeTransl{example}{Name-sg}{Example}
2370 \zcDicTypeTransl{example}{name-sg}{example}
2371 \zcDicTypeTransl{example}{Name-pl}{Examples}
2372 \zcDicTypeTransl{example}{name-pl}{examples}
2373
2374 \zcDicTypeTransl{remark}{Name-sg}{Remark}
2375 \zcDicTypeTransl{remark}{name-sg}{remark}
2376 \zcDicTypeTransl{remark}{Name-pl}{Remarks}
2377 \zcDicTypeTransl{remark}{name-pl}{remarks}
2378
2379 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithm}
2380 \zcDicTypeTransl{algorithm}{name-sg}{algorithm}
2381 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithms}
2382 \zcDicTypeTransl{algorithm}{name-pl}{algorithms}
2383
2384 \zcDicTypeTransl{listing}{Name-sg}{Listing}
2385 \zcDicTypeTransl{listing}{name-sg}{listing}
2386 \zcDicTypeTransl{listing}{Name-pl}{Listings}
2387 \zcDicTypeTransl{listing}{name-pl}{listings}
2388
2389 \zcDicTypeTransl{exercise}{Name-sg}{Exercise}
2390 \zcDicTypeTransl{exercise}{name-sg}{exercise}
2391 \zcDicTypeTransl{exercise}{Name-pl}{Exercises}
2392 \zcDicTypeTransl{exercise}{name-pl}{exercises}
2393
2394 \zcDicTypeTransl{solution}{Name-sg}{Solution}
2395 \zcDicTypeTransl{solution}{name-sg}{solution}
2396 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
2397 \zcDicTypeTransl{solution}{name-pl}{solutions}
2398 </lang-english>
2399 <*lang-german>

```

German

```

2400 \ProvideDictionaryFor{German}{zref-clever}
2401
2402 \zcDicDefaultTransl{namesep}{\nobreakspace}
2403 \zcDicDefaultTransl{pairsep}{\~und\nobreakspace}

```

2404 \zcDicDefaultTransl{listsep}{~,~}
 2405 \zcDicDefaultTransl{lastsep}{~und\nobreakspace}
 2406 \zcDicDefaultTransl{tpairsep}{~und\nobreakspace}
 2407 \zcDicDefaultTransl{tlistsep}{~,~}
 2408 \zcDicDefaultTransl{tlastsep}{~und\nobreakspace}
 2409 \zcDicDefaultTransl{notesep}{~}
 2410 \zcDicDefaultTransl{rangesep}{~bis\nobreakspace}
 2411
 2412 \zcDicTypeTransl{part}{Name-sg}{Teil}
 2413 \zcDicTypeTransl{part}{name-sg}{Teil}
 2414 \zcDicTypeTransl{part}{Name-pl}{Teile}
 2415 \zcDicTypeTransl{part}{name-pl}{Teile}
 2416
 2417 \zcDicTypeTransl{chapter}{Name-sg}{Kapitel}
 2418 \zcDicTypeTransl{chapter}{name-sg}{Kapitel}
 2419 \zcDicTypeTransl{chapter}{Name-pl}{Kapitel}
 2420 \zcDicTypeTransl{chapter}{name-pl}{Kapitel}
 2421
 2422 \zcDicTypeTransl{section}{Name-sg}{Abschnitt}
 2423 \zcDicTypeTransl{section}{name-sg}{Abschnitt}
 2424 \zcDicTypeTransl{section}{Name-pl}{Abschnitte}
 2425 \zcDicTypeTransl{section}{name-pl}{Abschnitte}
 2426
 2427 \zcDicTypeTransl{paragraph}{Name-sg}{Absatz}
 2428 \zcDicTypeTransl{paragraph}{name-sg}{Absatz}
 2429 \zcDicTypeTransl{paragraph}{Name-pl}{Absätze}
 2430 \zcDicTypeTransl{paragraph}{name-pl}{Absätze}
 2431
 2432 \zcDicTypeTransl{appendix}{Name-sg}{Anhang}
 2433 \zcDicTypeTransl{appendix}{name-sg}{Anhang}
 2434 \zcDicTypeTransl{appendix}{Name-pl}{Anhänge}
 2435 \zcDicTypeTransl{appendix}{name-pl}{Anhänge}
 2436
 2437 \zcDicTypeTransl{page}{Name-sg}{Seite}
 2438 \zcDicTypeTransl{page}{name-sg}{Seite}
 2439 \zcDicTypeTransl{page}{Name-pl}{Seiten}
 2440 \zcDicTypeTransl{page}{name-pl}{Seiten}
 2441
 2442 \zcDicTypeTransl{line}{Name-sg}{Zeile}
 2443 \zcDicTypeTransl{line}{name-sg}{Zeile}
 2444 \zcDicTypeTransl{line}{Name-pl}{Zeilen}
 2445 \zcDicTypeTransl{line}{name-pl}{Zeilen}
 2446
 2447 \zcDicTypeTransl{figure}{Name-sg}{Abbildung}
 2448 \zcDicTypeTransl{figure}{name-sg}{Abbildung}
 2449 \zcDicTypeTransl{figure}{Name-pl}{Abbildungen}
 2450 \zcDicTypeTransl{figure}{name-pl}{Abbildungen}
 2451 \zcDicTypeTransl{figure}{Name-sg-ab}{Abb.}
 2452 \zcDicTypeTransl{figure}{name-sg-ab}{Abb.}
 2453 \zcDicTypeTransl{figure}{Name-pl-ab}{Abb.}
 2454 \zcDicTypeTransl{figure}{name-pl-ab}{Abb.}
 2455
 2456 \zcDicTypeTransl{table}{Name-sg}{Tabelle}
 2457 \zcDicTypeTransl{table}{name-sg}{Tabelle}

2458 \zcDicTypeTransl{table}{Name-pl}{Tabellen}
 2459 \zcDicTypeTransl{table}{name-pl}{Tabellen}
 2460
 2461 \zcDicTypeTransl{item}{Name-sg}{Punkt}
 2462 \zcDicTypeTransl{item}{name-sg}{Punkt}
 2463 \zcDicTypeTransl{item}{Name-pl}{Punkte}
 2464 \zcDicTypeTransl{item}{name-pl}{Punkte}
 2465
 2466 \zcDicTypeTransl{footnote}{Name-sg}{Fußnote}
 2467 \zcDicTypeTransl{footnote}{name-sg}{Fußnote}
 2468 \zcDicTypeTransl{footnote}{Name-pl}{Fußnoten}
 2469 \zcDicTypeTransl{footnote}{name-pl}{Fußnoten}
 2470
 2471 \zcDicTypeTransl{note}{Name-sg}{Anmerkung}
 2472 \zcDicTypeTransl{note}{name-sg}{Anmerkung}
 2473 \zcDicTypeTransl{note}{Name-pl}{Anmerkungen}
 2474 \zcDicTypeTransl{note}{name-pl}{Anmerkungen}
 2475
 2476 \zcDicTypeTransl{equation}{Name-sg}{Gleichung}
 2477 \zcDicTypeTransl{equation}{name-sg}{Gleichung}
 2478 \zcDicTypeTransl{equation}{Name-pl}{Gleichungen}
 2479 \zcDicTypeTransl{equation}{name-pl}{Gleichungen}
 2480 \zcDicTypeTransl{equation}{refpre-in}{(}
 2481 \zcDicTypeTransl{equation}{refpos-in}{)}
 2482
 2483 \zcDicTypeTransl{theorem}{Name-sg}{Theorem}
 2484 \zcDicTypeTransl{theorem}{name-sg}{Theorem}
 2485 \zcDicTypeTransl{theorem}{Name-pl}{Theoreme}
 2486 \zcDicTypeTransl{theorem}{name-pl}{Theoreme}
 2487
 2488 \zcDicTypeTransl{lemma}{Name-sg}{Lemma}
 2489 \zcDicTypeTransl{lemma}{name-sg}{Lemma}
 2490 \zcDicTypeTransl{lemma}{Name-pl}{Lemmata}
 2491 \zcDicTypeTransl{lemma}{name-pl}{Lemmata}
 2492
 2493 \zcDicTypeTransl{corollary}{Name-sg}{Korollar}
 2494 \zcDicTypeTransl{corollary}{name-sg}{Korollar}
 2495 \zcDicTypeTransl{corollary}{Name-pl}{Korollare}
 2496 \zcDicTypeTransl{corollary}{name-pl}{Korollare}
 2497
 2498 \zcDicTypeTransl{proposition}{Name-sg}{Satz}
 2499 \zcDicTypeTransl{proposition}{name-sg}{Satz}
 2500 \zcDicTypeTransl{proposition}{Name-pl}{Sätze}
 2501 \zcDicTypeTransl{proposition}{name-pl}{Sätze}
 2502
 2503 \zcDicTypeTransl{definition}{Name-sg}{Definition}
 2504 \zcDicTypeTransl{definition}{name-sg}{Definition}
 2505 \zcDicTypeTransl{definition}{Name-pl}{Definitionen}
 2506 \zcDicTypeTransl{definition}{name-pl}{Definitionen}
 2507
 2508 \zcDicTypeTransl{proof}{Name-sg}{Beweis}
 2509 \zcDicTypeTransl{proof}{name-sg}{Beweis}
 2510 \zcDicTypeTransl{proof}{Name-pl}{Beweise}
 2511 \zcDicTypeTransl{proof}{name-pl}{Beweise}

```

2512
2513 \zcDicTypeTransl{result}{Name-sg}{Ergebnis}
2514 \zcDicTypeTransl{result}{name-sg}{Ergebnis}
2515 \zcDicTypeTransl{result}{Name-pl}{Ergebnisse}
2516 \zcDicTypeTransl{result}{name-pl}{Ergebnisse}
2517
2518 \zcDicTypeTransl{example}{Name-sg}{Beispiel}
2519 \zcDicTypeTransl{example}{name-sg}{Beispiel}
2520 \zcDicTypeTransl{example}{Name-pl}{Beispiele}
2521 \zcDicTypeTransl{example}{name-pl}{Beispiele}
2522
2523 \zcDicTypeTransl{remark}{Name-sg}{Bemerkung}
2524 \zcDicTypeTransl{remark}{name-sg}{Bemerkung}
2525 \zcDicTypeTransl{remark}{Name-pl}{Bemerkungen}
2526 \zcDicTypeTransl{remark}{name-pl}{Bemerkungen}
2527
2528 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithmus}
2529 \zcDicTypeTransl{algorithm}{name-sg}{Algorithmus}
2530 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithmen}
2531 \zcDicTypeTransl{algorithm}{name-pl}{Algorithmen}
2532
2533 \zcDicTypeTransl{listing}{Name-sg}{Listing} % CHECK
2534 \zcDicTypeTransl{listing}{name-sg}{Listing} % CHECK
2535 \zcDicTypeTransl{listing}{Name-pl}{Listings} % CHECK
2536 \zcDicTypeTransl{listing}{name-pl}{Listings} % CHECK
2537
2538 \zcDicTypeTransl{exercise}{Name-sg}{Übungsaufgabe}
2539 \zcDicTypeTransl{exercise}{name-sg}{Übungsaufgabe}
2540 \zcDicTypeTransl{exercise}{Name-pl}{Übungsaufgaben}
2541 \zcDicTypeTransl{exercise}{name-pl}{Übungsaufgaben}
2542
2543 \zcDicTypeTransl{solution}{Name-sg}{Lösung}
2544 \zcDicTypeTransl{solution}{name-sg}{Lösung}
2545 \zcDicTypeTransl{solution}{Name-pl}{Lösungen}
2546 \zcDicTypeTransl{solution}{name-pl}{Lösungen}
2547 </lang-german>
2548 <*lang-french>

```

French

```

2549 \ProvideDictionaryFor{French}{zref-clever}
2550
2551 \zcDicDefaultTransl{namesep}{\nobreakspace}
2552 \zcDicDefaultTransl{pairsep}{~et\nobreakspace}
2553 \zcDicDefaultTransl{listsep}{~,~}
2554 \zcDicDefaultTransl{lastsep}{~et\nobreakspace}
2555 \zcDicDefaultTransl{tpairsep}{~et\nobreakspace}
2556 \zcDicDefaultTransl{tlistsep}{~,~}
2557 \zcDicDefaultTransl{tlastsep}{~et\nobreakspace}
2558 \zcDicDefaultTransl{notesep}{~}
2559 \zcDicDefaultTransl{rangesep}{~à\nobreakspace}
2560
2561 \zcDicTypeTransl{part}{Name-sg}{Partie}
2562 \zcDicTypeTransl{part}{name-sg}{partie}

```

2563 \zcDicTypeTransl{part}{Name-pl}{Parties}
 2564 \zcDicTypeTransl{part}{name-pl}{parties}
 2565
 2566 \zcDicTypeTransl{chapter}{Name-sg}{Chapitre}
 2567 \zcDicTypeTransl{chapter}{name-sg}{chapitre}
 2568 \zcDicTypeTransl{chapter}{Name-pl}{Chapitres}
 2569 \zcDicTypeTransl{chapter}{name-pl}{chapitres}
 2570
 2571 \zcDicTypeTransl{section}{Name-sg}{Section}
 2572 \zcDicTypeTransl{section}{name-sg}{section}
 2573 \zcDicTypeTransl{section}{Name-pl}{Sections}
 2574 \zcDicTypeTransl{section}{name-pl}{sections}
 2575
 2576 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraphe}
 2577 \zcDicTypeTransl{paragraph}{name-sg}{paragraphe}
 2578 \zcDicTypeTransl{paragraph}{Name-pl}{Paragraphes}
 2579 \zcDicTypeTransl{paragraph}{name-pl}{paragraphes}
 2580
 2581 \zcDicTypeTransl{appendix}{Name-sg}{Annexe}
 2582 \zcDicTypeTransl{appendix}{name-sg}{annexe}
 2583 \zcDicTypeTransl{appendix}{Name-pl}{Annexes}
 2584 \zcDicTypeTransl{appendix}{name-pl}{annexes}
 2585
 2586 \zcDicTypeTransl{page}{Name-sg}{Page}
 2587 \zcDicTypeTransl{page}{name-sg}{page}
 2588 \zcDicTypeTransl{page}{Name-pl}{Pages}
 2589 \zcDicTypeTransl{page}{name-pl}{pages}
 2590
 2591 \zcDicTypeTransl{line}{Name-sg}{Ligne}
 2592 \zcDicTypeTransl{line}{name-sg}{ligne}
 2593 \zcDicTypeTransl{line}{Name-pl}{Lignes}
 2594 \zcDicTypeTransl{line}{name-pl}{lignes}
 2595
 2596 \zcDicTypeTransl{figure}{Name-sg}{Figure}
 2597 \zcDicTypeTransl{figure}{name-sg}{figure}
 2598 \zcDicTypeTransl{figure}{Name-pl}{Figures}
 2599 \zcDicTypeTransl{figure}{name-pl}{figures}
 2600
 2601 \zcDicTypeTransl{table}{Name-sg}{Table}
 2602 \zcDicTypeTransl{table}{name-sg}{table}
 2603 \zcDicTypeTransl{table}{Name-pl}{Tables}
 2604 \zcDicTypeTransl{table}{name-pl}{tables}
 2605
 2606 \zcDicTypeTransl{item}{Name-sg}{Point}
 2607 \zcDicTypeTransl{item}{name-sg}{point}
 2608 \zcDicTypeTransl{item}{Name-pl}{Points}
 2609 \zcDicTypeTransl{item}{name-pl}{points}
 2610
 2611 \zcDicTypeTransl{footnote}{Name-sg}{Note}
 2612 \zcDicTypeTransl{footnote}{name-sg}{note}
 2613 \zcDicTypeTransl{footnote}{Name-pl}{Notes}
 2614 \zcDicTypeTransl{footnote}{name-pl}{notes}
 2615
 2616 \zcDicTypeTransl{note}{Name-sg}{Note}

2617 \zcDicTypeTransl{note}{name-sg}{note}
 2618 \zcDicTypeTransl{note}{Name-pl}{Notes}
 2619 \zcDicTypeTransl{note}{name-pl}{notes}
 2620
 2621 \zcDicTypeTransl{equation}{Name-sg}{Équation}
 2622 \zcDicTypeTransl{equation}{name-sg}{équation}
 2623 \zcDicTypeTransl{equation}{Name-pl}{Équations}
 2624 \zcDicTypeTransl{equation}{name-pl}{équations}
 2625 \zcDicTypeTransl{equation}{refpre-in}{(}
 2626 \zcDicTypeTransl{equation}{refpos-in}{)}
 2627
 2628 \zcDicTypeTransl{theorem}{Name-sg}{Théorème}
 2629 \zcDicTypeTransl{theorem}{name-sg}{théorème}
 2630 \zcDicTypeTransl{theorem}{Name-pl}{Théorèmes}
 2631 \zcDicTypeTransl{theorem}{name-pl}{théorèmes}
 2632
 2633 \zcDicTypeTransl{lemma}{Name-sg}{Lemme}
 2634 \zcDicTypeTransl{lemma}{name-sg}{lemme}
 2635 \zcDicTypeTransl{lemma}{Name-pl}{Lemmes}
 2636 \zcDicTypeTransl{lemma}{name-pl}{lemmes}
 2637
 2638 \zcDicTypeTransl{corollary}{Name-sg}{Corollaire}
 2639 \zcDicTypeTransl{corollary}{name-sg}{corollaire}
 2640 \zcDicTypeTransl{corollary}{Name-pl}{Corollaires}
 2641 \zcDicTypeTransl{corollary}{name-pl}{corollaires}
 2642
 2643 \zcDicTypeTransl{proposition}{Name-sg}{Proposition}
 2644 \zcDicTypeTransl{proposition}{name-sg}{proposition}
 2645 \zcDicTypeTransl{proposition}{Name-pl}{Propositions}
 2646 \zcDicTypeTransl{proposition}{name-pl}{propositions}
 2647
 2648 \zcDicTypeTransl{definition}{Name-sg}{Définition}
 2649 \zcDicTypeTransl{definition}{name-sg}{définition}
 2650 \zcDicTypeTransl{definition}{Name-pl}{Définitions}
 2651 \zcDicTypeTransl{definition}{name-pl}{définitions}
 2652
 2653 \zcDicTypeTransl{proof}{Name-sg}{Démonstration}
 2654 \zcDicTypeTransl{proof}{name-sg}{démonstration}
 2655 \zcDicTypeTransl{proof}{Name-pl}{Démonstrations}
 2656 \zcDicTypeTransl{proof}{name-pl}{démonstrations}
 2657
 2658 \zcDicTypeTransl{result}{Name-sg}{Résultat}
 2659 \zcDicTypeTransl{result}{name-sg}{résultat}
 2660 \zcDicTypeTransl{result}{Name-pl}{Résultats}
 2661 \zcDicTypeTransl{result}{name-pl}{résultats}
 2662
 2663 \zcDicTypeTransl{example}{Name-sg}{Exemple}
 2664 \zcDicTypeTransl{example}{name-sg}{exemple}
 2665 \zcDicTypeTransl{example}{Name-pl}{Exemples}
 2666 \zcDicTypeTransl{example}{name-pl}{exemples}
 2667
 2668 \zcDicTypeTransl{remark}{Name-sg}{Remarque}
 2669 \zcDicTypeTransl{remark}{name-sg}{remarque}
 2670 \zcDicTypeTransl{remark}{Name-pl}{Remarques}

2671 \zcDicTypeTransl{remark}{name-pl}{remarques}
 2672
 2673 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithme}
 2674 \zcDicTypeTransl{algorithm}{name-sg}{algorithme}
 2675 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithmes}
 2676 \zcDicTypeTransl{algorithm}{name-pl}{algorithmes}
 2677
 2678 \zcDicTypeTransl{listing}{Name-sg}{Liste}
 2679 \zcDicTypeTransl{listing}{name-sg}{liste}
 2680 \zcDicTypeTransl{listing}{Name-pl}{Listes}
 2681 \zcDicTypeTransl{listing}{name-pl}{listes}
 2682
 2683 \zcDicTypeTransl{exercise}{Name-sg}{Exercice}
 2684 \zcDicTypeTransl{exercise}{name-sg}{exercice}
 2685 \zcDicTypeTransl{exercise}{Name-pl}{Exercices}
 2686 \zcDicTypeTransl{exercise}{name-pl}{exercices}
 2687
 2688 \zcDicTypeTransl{solution}{Name-sg}{Solution}
 2689 \zcDicTypeTransl{solution}{name-sg}{solution}
 2690 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
 2691 \zcDicTypeTransl{solution}{name-pl}{solutions}
 2692 \langle /lang-french \rangle
 2693 \langle *lang-portuguese \rangle

Portuguese

2694 \ProvideDictionaryFor{Portuguese}{zref-clever}
 2695
 2696 \zcDicDefaultTransl{namesep}{\nobreakspace}
 2697 \zcDicDefaultTransl{pairsep}{~e\nobreakspace}
 2698 \zcDicDefaultTransl{listsep}{~,~}
 2699 \zcDicDefaultTransl{lastsep}{~e\nobreakspace}
 2700 \zcDicDefaultTransl{tpairsep}{~e\nobreakspace}
 2701 \zcDicDefaultTransl{tlistsep}{~,~}
 2702 \zcDicDefaultTransl{tlastsep}{~e\nobreakspace}
 2703 \zcDicDefaultTransl{notesep}{~}
 2704 \zcDicDefaultTransl{rangesep}{~a\nobreakspace}
 2705
 2706 \zcDicTypeTransl{part}{Name-sg}{Parte}
 2707 \zcDicTypeTransl{part}{name-sg}{parte}
 2708 \zcDicTypeTransl{part}{Name-pl}{Partes}
 2709 \zcDicTypeTransl{part}{name-pl}{partes}
 2710
 2711 \zcDicTypeTransl{chapter}{Name-sg}{Capítulo}
 2712 \zcDicTypeTransl{chapter}{name-sg}{capítulo}
 2713 \zcDicTypeTransl{chapter}{Name-pl}{Capítulos}
 2714 \zcDicTypeTransl{chapter}{name-pl}{capítulos}
 2715
 2716 \zcDicTypeTransl{section}{Name-sg}{Seção}
 2717 \zcDicTypeTransl{section}{name-sg}{seção}
 2718 \zcDicTypeTransl{section}{Name-pl}{Seções}
 2719 \zcDicTypeTransl{section}{name-pl}{seções}
 2720
 2721 \zcDicTypeTransl{paragraph}{Name-sg}{Parágrafo}

2722 \zcDicTypeTransl{paragraph}{name-sg}{parágrafo}
 2723 \zcDicTypeTransl{paragraph}{Name-pl}{Parágrafos}
 2724 \zcDicTypeTransl{paragraph}{name-pl}{parágrafos}
 2725 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}
 2726 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}
 2727 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}
 2728 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}
 2729
 2730 \zcDicTypeTransl{appendix}{Name-sg}{Apêndice}
 2731 \zcDicTypeTransl{appendix}{name-sg}{apêndice}
 2732 \zcDicTypeTransl{appendix}{Name-pl}{Apêndices}
 2733 \zcDicTypeTransl{appendix}{name-pl}{apêndices}
 2734
 2735 \zcDicTypeTransl{page}{Name-sg}{Página}
 2736 \zcDicTypeTransl{page}{name-sg}{página}
 2737 \zcDicTypeTransl{page}{Name-pl}{Páginas}
 2738 \zcDicTypeTransl{page}{name-pl}{páginas}
 2739 \zcDicTypeTransl{page}{name-sg-ab}{p.}
 2740 \zcDicTypeTransl{page}{name-pl-ab}{pp.}
 2741
 2742 \zcDicTypeTransl{line}{Name-sg}{Linha}
 2743 \zcDicTypeTransl{line}{name-sg}{linha}
 2744 \zcDicTypeTransl{line}{Name-pl}{Linhas}
 2745 \zcDicTypeTransl{line}{name-pl}{linhas}
 2746
 2747 \zcDicTypeTransl{figure}{Name-sg}{Figura}
 2748 \zcDicTypeTransl{figure}{name-sg}{figura}
 2749 \zcDicTypeTransl{figure}{Name-pl}{Figuras}
 2750 \zcDicTypeTransl{figure}{name-pl}{figuras}
 2751 \zcDicTypeTransl{figure}{Name-sg-ab}{Fig.}
 2752 \zcDicTypeTransl{figure}{name-sg-ab}{fig.}
 2753 \zcDicTypeTransl{figure}{Name-pl-ab}{Figs.}
 2754 \zcDicTypeTransl{figure}{name-pl-ab}{figs.}
 2755
 2756 \zcDicTypeTransl{table}{Name-sg}{Tabela}
 2757 \zcDicTypeTransl{table}{name-sg}{tabela}
 2758 \zcDicTypeTransl{table}{Name-pl}{Tabelas}
 2759 \zcDicTypeTransl{table}{name-pl}{tabelas}
 2760
 2761 \zcDicTypeTransl{item}{Name-sg}{Item}
 2762 \zcDicTypeTransl{item}{name-sg}{item}
 2763 \zcDicTypeTransl{item}{Name-pl}{Itens}
 2764 \zcDicTypeTransl{item}{name-pl}{itens}
 2765
 2766 \zcDicTypeTransl{footnote}{Name-sg}{Nota}
 2767 \zcDicTypeTransl{footnote}{name-sg}{nota}
 2768 \zcDicTypeTransl{footnote}{Name-pl}{Notas}
 2769 \zcDicTypeTransl{footnote}{name-pl}{notas}
 2770
 2771 \zcDicTypeTransl{note}{Name-sg}{Nota}
 2772 \zcDicTypeTransl{note}{name-sg}{nota}
 2773 \zcDicTypeTransl{note}{Name-pl}{Notas}
 2774 \zcDicTypeTransl{note}{name-pl}{notas}
 2775

2776 \zcDicTypeTransl{equation}{Name-sg}{Equação}
 2777 \zcDicTypeTransl{equation}{name-sg}{equação}
 2778 \zcDicTypeTransl{equation}{Name-pl}{Equações}
 2779 \zcDicTypeTransl{equation}{name-pl}{equações}
 2780 \zcDicTypeTransl{equation}{Name-sg-ab}{Eq.}
 2781 \zcDicTypeTransl{equation}{name-sg-ab}{eq.}
 2782 \zcDicTypeTransl{equation}{Name-pl-ab}{Eqs.}
 2783 \zcDicTypeTransl{equation}{name-pl-ab}{eqs.}
 2784 \zcDicTypeTransl{equation}{refpre-in}{(
 2785 \zcDicTypeTransl{equation}{refpos-in}{)})
 2786
 2787 \zcDicTypeTransl{theorem}{Name-sg}{Teorema}
 2788 \zcDicTypeTransl{theorem}{name-sg}{teorema}
 2789 \zcDicTypeTransl{theorem}{Name-pl}{Teoremas}
 2790 \zcDicTypeTransl{theorem}{name-pl}{teoremas}
 2791
 2792 \zcDicTypeTransl{lemma}{Name-sg}{Lema}
 2793 \zcDicTypeTransl{lemma}{name-sg}{lema}
 2794 \zcDicTypeTransl{lemma}{Name-pl}{Lemas}
 2795 \zcDicTypeTransl{lemma}{name-pl}{lemas}
 2796
 2797 \zcDicTypeTransl{corollary}{Name-sg}{Corolário}
 2798 \zcDicTypeTransl{corollary}{name-sg}{corolário}
 2799 \zcDicTypeTransl{corollary}{Name-pl}{Corolários}
 2800 \zcDicTypeTransl{corollary}{name-pl}{corolários}
 2801
 2802 \zcDicTypeTransl{proposition}{Name-sg}{Proposição}
 2803 \zcDicTypeTransl{proposition}{name-sg}{proposição}
 2804 \zcDicTypeTransl{proposition}{Name-pl}{Proposições}
 2805 \zcDicTypeTransl{proposition}{name-pl}{proposições}
 2806
 2807 \zcDicTypeTransl{definition}{Name-sg}{Definição}
 2808 \zcDicTypeTransl{definition}{name-sg}{definição}
 2809 \zcDicTypeTransl{definition}{Name-pl}{Definições}
 2810 \zcDicTypeTransl{definition}{name-pl}{definições}
 2811
 2812 \zcDicTypeTransl{proof}{Name-sg}{Demonstração}
 2813 \zcDicTypeTransl{proof}{name-sg}{demonstração}
 2814 \zcDicTypeTransl{proof}{Name-pl}{Demonstrações}
 2815 \zcDicTypeTransl{proof}{name-pl}{demonstrações}
 2816
 2817 \zcDicTypeTransl{result}{Name-sg}{Resultado}
 2818 \zcDicTypeTransl{result}{name-sg}{resultado}
 2819 \zcDicTypeTransl{result}{Name-pl}{Resultados}
 2820 \zcDicTypeTransl{result}{name-pl}{resultados}
 2821
 2822 \zcDicTypeTransl{example}{Name-sg}{Exemplo}
 2823 \zcDicTypeTransl{example}{name-sg}{exemplo}
 2824 \zcDicTypeTransl{example}{Name-pl}{Exemplos}
 2825 \zcDicTypeTransl{example}{name-pl}{exemplos}
 2826
 2827 \zcDicTypeTransl{remark}{Name-sg}{Observação}
 2828 \zcDicTypeTransl{remark}{name-sg}{observação}
 2829 \zcDicTypeTransl{remark}{Name-pl}{Observações}

2830 \zcDicTypeTransl{remark}{name-pl}{observações}
 2831
 2832 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}
 2833 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}
 2834 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}
 2835 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}
 2836
 2837 \zcDicTypeTransl{listing}{Name-sg}{Listagem}
 2838 \zcDicTypeTransl{listing}{name-sg}{listagem}
 2839 \zcDicTypeTransl{listing}{Name-pl}{Listagens}
 2840 \zcDicTypeTransl{listing}{name-pl}{listagens}
 2841
 2842 \zcDicTypeTransl{exercise}{Name-sg}{Exercício}
 2843 \zcDicTypeTransl{exercise}{name-sg}{exercício}
 2844 \zcDicTypeTransl{exercise}{Name-pl}{Exercícios}
 2845 \zcDicTypeTransl{exercise}{name-pl}{exercícios}
 2846
 2847 \zcDicTypeTransl{solution}{Name-sg}{Solução}
 2848 \zcDicTypeTransl{solution}{name-sg}{solução}
 2849 \zcDicTypeTransl{solution}{Name-pl}{Soluções}
 2850 \zcDicTypeTransl{solution}{name-pl}{soluções}
 2851 \langle /lang-portuguese \rangle
 2852 \langle *lang-spanish \rangle

Spanish

2853 \ProvideDictionaryFor{Spanish}{zref-clever}
 2854
 2855 \zcDicDefaultTransl{namesep}{\nobreakspace}
 2856 \zcDicDefaultTransl{pairsep}{~y\nobreakspace}
 2857 \zcDicDefaultTransl{listsep}{~,~}
 2858 \zcDicDefaultTransl{lastsep}{~y\nobreakspace}
 2859 \zcDicDefaultTransl{tpairsep}{~y\nobreakspace}
 2860 \zcDicDefaultTransl{tlistsep}{~,~}
 2861 \zcDicDefaultTransl{tlastsep}{~y\nobreakspace}
 2862 \zcDicDefaultTransl{notesep}{~}
 2863 \zcDicDefaultTransl{rangesep}{~a\nobreakspace}
 2864
 2865 \zcDicTypeTransl{part}{Name-sg}{Parte}
 2866 \zcDicTypeTransl{part}{name-sg}{parte}
 2867 \zcDicTypeTransl{part}{Name-pl}{Partes}
 2868 \zcDicTypeTransl{part}{name-pl}{partes}
 2869
 2870 \zcDicTypeTransl{chapter}{Name-sg}{Capítulo}
 2871 \zcDicTypeTransl{chapter}{name-sg}{capítulo}
 2872 \zcDicTypeTransl{chapter}{Name-pl}{Capítulos}
 2873 \zcDicTypeTransl{chapter}{name-pl}{capítulos}
 2874
 2875 \zcDicTypeTransl{section}{Name-sg}{Sección}
 2876 \zcDicTypeTransl{section}{name-sg}{sección}
 2877 \zcDicTypeTransl{section}{Name-pl}{Secciones}
 2878 \zcDicTypeTransl{section}{name-pl}{secciones}
 2879
 2880 \zcDicTypeTransl{paragraph}{Name-sg}{Párrafo}

2881 \zcDicTypeTransl{paragraph}{name-sg}{párrafo}
 2882 \zcDicTypeTransl{paragraph}{Name-pl}{Párrafos}
 2883 \zcDicTypeTransl{paragraph}{name-pl}{párrafos}
 2884
 2885 \zcDicTypeTransl{appendix}{Name-sg}{Apéndice}
 2886 \zcDicTypeTransl{appendix}{name-sg}{apéndice}
 2887 \zcDicTypeTransl{appendix}{Name-pl}{Apéndices}
 2888 \zcDicTypeTransl{appendix}{name-pl}{apéndices}
 2889
 2890 \zcDicTypeTransl{page}{Name-sg}{Página}
 2891 \zcDicTypeTransl{page}{name-sg}{página}
 2892 \zcDicTypeTransl{page}{Name-pl}{Páginas}
 2893 \zcDicTypeTransl{page}{name-pl}{páginas}
 2894
 2895 \zcDicTypeTransl{line}{Name-sg}{Línea}
 2896 \zcDicTypeTransl{line}{name-sg}{línea}
 2897 \zcDicTypeTransl{line}{Name-pl}{Líneas}
 2898 \zcDicTypeTransl{line}{name-pl}{líneas}
 2899
 2900 \zcDicTypeTransl{figure}{Name-sg}{Figura}
 2901 \zcDicTypeTransl{figure}{name-sg}{figura}
 2902 \zcDicTypeTransl{figure}{Name-pl}{Figuras}
 2903 \zcDicTypeTransl{figure}{name-pl}{figuras}
 2904
 2905 \zcDicTypeTransl{table}{Name-sg}{Cuadro}
 2906 \zcDicTypeTransl{table}{name-sg}{cuadro}
 2907 \zcDicTypeTransl{table}{Name-pl}{Cuadros}
 2908 \zcDicTypeTransl{table}{name-pl}{cuadros}
 2909
 2910 \zcDicTypeTransl{item}{Name-sg}{Punto}
 2911 \zcDicTypeTransl{item}{name-sg}{punto}
 2912 \zcDicTypeTransl{item}{Name-pl}{Puntos}
 2913 \zcDicTypeTransl{item}{name-pl}{puntos}
 2914
 2915 \zcDicTypeTransl{footnote}{Name-sg}{Nota}
 2916 \zcDicTypeTransl{footnote}{name-sg}{nota}
 2917 \zcDicTypeTransl{footnote}{Name-pl}{Notas}
 2918 \zcDicTypeTransl{footnote}{name-pl}{notas}
 2919
 2920 \zcDicTypeTransl{note}{Name-sg}{Nota}
 2921 \zcDicTypeTransl{note}{name-sg}{nota}
 2922 \zcDicTypeTransl{note}{Name-pl}{Notas}
 2923 \zcDicTypeTransl{note}{name-pl}{notas}
 2924
 2925 \zcDicTypeTransl{equation}{Name-sg}{Ecuación}
 2926 \zcDicTypeTransl{equation}{name-sg}{ecuación}
 2927 \zcDicTypeTransl{equation}{Name-pl}{Ecuaciones}
 2928 \zcDicTypeTransl{equation}{name-pl}{ecuaciones}
 2929 \zcDicTypeTransl{equation}{refpre-in}{(}
 2930 \zcDicTypeTransl{equation}{refpos-in}{)}
 2931
 2932 \zcDicTypeTransl{theorem}{Name-sg}{Teorema}
 2933 \zcDicTypeTransl{theorem}{name-sg}{teorema}
 2934 \zcDicTypeTransl{theorem}{Name-pl}{Teoremas}

2935 \zcDicTypeTransl{theorem}{name-pl}{teoremas}
 2936
 2937 \zcDicTypeTransl{lemma}{Name-sg}{Lema}
 2938 \zcDicTypeTransl{lemma}{name-sg}{lema}
 2939 \zcDicTypeTransl{lemma}{Name-pl}{Lemas}
 2940 \zcDicTypeTransl{lemma}{name-pl}{lemas}
 2941
 2942 \zcDicTypeTransl{corollary}{Name-sg}{Corolario}
 2943 \zcDicTypeTransl{corollary}{name-sg}{corolario}
 2944 \zcDicTypeTransl{corollary}{Name-pl}{Corolarios}
 2945 \zcDicTypeTransl{corollary}{name-pl}{corolarios}
 2946
 2947 \zcDicTypeTransl{proposition}{Name-sg}{Proposición}
 2948 \zcDicTypeTransl{proposition}{name-sg}{proposición}
 2949 \zcDicTypeTransl{proposition}{Name-pl}{Proposiciones}
 2950 \zcDicTypeTransl{proposition}{name-pl}{proposiciones}
 2951
 2952 \zcDicTypeTransl{definition}{Name-sg}{Definición}
 2953 \zcDicTypeTransl{definition}{name-sg}{definición}
 2954 \zcDicTypeTransl{definition}{Name-pl}{Definiciones}
 2955 \zcDicTypeTransl{definition}{name-pl}{definiciones}
 2956
 2957 \zcDicTypeTransl{proof}{Name-sg}{Demostración}
 2958 \zcDicTypeTransl{proof}{name-sg}{demostración}
 2959 \zcDicTypeTransl{proof}{Name-pl}{Demostraciones}
 2960 \zcDicTypeTransl{proof}{name-pl}{demostraciones}
 2961
 2962 \zcDicTypeTransl{result}{Name-sg}{Resultado}
 2963 \zcDicTypeTransl{result}{name-sg}{resultado}
 2964 \zcDicTypeTransl{result}{Name-pl}{Resultados}
 2965 \zcDicTypeTransl{result}{name-pl}{resultados}
 2966
 2967 \zcDicTypeTransl{example}{Name-sg}{Ejemplo}
 2968 \zcDicTypeTransl{example}{name-sg}{ejemplo}
 2969 \zcDicTypeTransl{example}{Name-pl}{Ejemplos}
 2970 \zcDicTypeTransl{example}{name-pl}{ejemplos}
 2971
 2972 \zcDicTypeTransl{remark}{Name-sg}{Observación}
 2973 \zcDicTypeTransl{remark}{name-sg}{observación}
 2974 \zcDicTypeTransl{remark}{Name-pl}{Observaciones}
 2975 \zcDicTypeTransl{remark}{name-pl}{observaciones}
 2976
 2977 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}
 2978 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}
 2979 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}
 2980 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}
 2981
 2982 \zcDicTypeTransl{listing}{Name-sg}{Listado}
 2983 \zcDicTypeTransl{listing}{name-sg}{listado}
 2984 \zcDicTypeTransl{listing}{Name-pl}{Listados}
 2985 \zcDicTypeTransl{listing}{name-pl}{listados}
 2986
 2987 \zcDicTypeTransl{exercise}{Name-sg}{Ejercicio}
 2988 \zcDicTypeTransl{exercise}{name-sg}{ejercicio}

```

2989 \zcDicTypeTransl{exercise}{Name-pl}{Ejercicios}
2990 \zcDicTypeTransl{exercise}{name-pl}{ejercicios}
2991
2992 \zcDicTypeTransl{solution}{Name-sg}{Solución}
2993 \zcDicTypeTransl{solution}{name-sg}{solución}
2994 \zcDicTypeTransl{solution}{Name-pl}{Soluciones}
2995 \zcDicTypeTransl{solution}{name-pl}{soluciones}
2996 </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		<code>\bool_until_do:Nn</code> 1008, 1256
<code>\AddToHook</code>		
. 91, 255, 270, 380, 453, 487, 513, 570		
<code>\appendix</code>	54	C
<code>\appendixname</code>	55	clist commands:
<code>\Arg</code>	2103	<code>\clist_map_inline:Nn</code> 521, 538
<code>\AtEndOfPackage</code>	16, 485	<code>\clist_map_inline:nn</code>
	 200, 597, 648, 664, 728, 753, 783
B		<code>\counterwithin</code>
<code>\baselanguage</code>	476, 528, 545	4
bool commands:		<code>\cs</code>
<code>\bool_case_true:</code>	2	cs commands:
<code>\bool_if:NTF</code>	384, 388,	<code>\cs_generate_variant:Nn</code>
1288, 1361, 1486, 1507, 1538, 1593,	 55, 56, 149, 152, 920, 1823
1660, 1664, 1671, 1680, 1686, 1967		<code>\cs_if_exist:NTF</code>
<code>\bool_if:nTF</code> ...	59, 928, 937, 946,	39, 48, 69
1015, 1043, 1066, 1155, 1163, 1302,		<code>\cs_new:Npn</code> 37, 46, 57, 67, 78, 1778, 1962
1310, 1519, 1526, 1533, 1782, 2045		<code>\cs_new_protected:Npn</code>
<code>\bool_lazy_all:nTF</code>	2110 147, 150, 153, 161, 816, 863,
<code>\bool_lazy_and:nnTF</code>		906, 921, 985, 1130, 1186, 1232,
.... 823, 836, 1635, 1844, 2155		1372, 1628, 1824, 2104, 2149, 2169
<code>\bool_lazy_any:nTF</code>	1935, 1944	<code>\cs_new_protected:Npx</code>
<code>\bool_lazy_or:nnTF</code>	827, 1832	90
<code>\bool_new:N</code>		<code>\cs_set_eq:NN</code>
. 291, 292, 317, 341, 350, 357, 358,		94
413, 414, 431, 432, 563, 564, 815,		
919, 1195, 1196, 1207, 1208, 1209, 1229		D
<code>\bool_set:Nn</code>	821	<code>\declaretranslation</code>
<code>\bool_set_false:N</code>		<code>\dots</code>
.... 304, 308, 365, 374, 375,		948, 958, 962
390, 585, 1004, 1255, 1294, 1308,		
1319, 1498, 1633, 1634, 1942, 1959		E
<code>\bool_set_true:N</code>		<code>\endinginput</code>
.... 298, 299, 303, 309, 364, 369,		12
370, 574, 579, 1027, 1038, 1055,		exp commands:
1061, 1078, 1084, 1110, 1122, 1262,		<code>\exp_args:NNe</code>
1289, 1295, 1299, 1320, 1323, 1958,		27
2181, 2188, 2189, 2207, 2214, 2215		<code>\exp_args:NNx</code>
		95, 1052, 1075
		<code>\exp_args:Nx</code> 471, 475, 523, 527, 540, 544
		<code>\exp_args:Nxx</code>
	 969, 1023, 2173, 2195, 2199
		<code>\exp_not:N</code>
		103,
		109, 1540, 1543, 1563, 1566, 1569,
		1785, 1788, 1791, 1803, 1805, 1808,
		1811, 1816, 1818, 1821, 1965, 1973,
		1991, 1994, 1996, 1999, 2005, 2012,

2014, 2018, 2021, 2024, 2026, 2032, 2036, 2039, 2051, 2054, 2057, 2080, 2082, 2085, 2088, 2094, 2096, 2099	\int_zero:N 1132, 1133, 1240, 1241, 1242, 1243, 1622, 1624, 1625, 1771, 1772
\exp_not:n . 1392, 1408, 1420, 1424, 1444, 1457, 1460, 1472, 1475, 1508, 1509, 1541, 1562, 1567, 1568, 1700, 1713, 1718, 1738, 1749, 1752, 1762, 1765, 1786, 1787, 1789, 1799, 1801, 1804, 1809, 1810, 1812, 1813, 1815, 1817, 1992, 1993, 1995, 1997, 1998, 2000, 2001, 2004, 2016, 2017, 2022, 2023, 2025, 2033, 2037, 2038, 2040, 2052, 2053, 2055, 2075, 2078, 2081, 2086, 2087, 2089, 2090, 2093, 2095	iow commands: \iow_newline: 120, 124
F	K
file commands:	keys commands:
\file_if_exist:nTF 471, 475, 523, 527, 540, 544	\keys_define:nn 21, 168, 196, 222, 246, 274, 281, 293, 318, 327, 342, 351, 359, 392, 399, 415, 433, 449, 489, 558, 565, 575, 586, 594, 619, 630, 656, 690, 715, 736, 766, 795
\fmtversion 3	\keys_set:nn 21, 580, 636, 646, 713, 819
G	keyval commands:
group commands:	\keyval_parse:nnn 172, 226
\group_begin: 93, 818, 1540, 1566, 1785, 1788, 1808, 1811, 1991, 1996, 1999, 2014, 2021, 2036, 2051, 2054, 2085, 2088	L
\group_end: 96, 844, 1563, 1569, 1803, 1805, 1816, 1818, 1994, 2005, 2012, 2018, 2024, 2039, 2080, 2082, 2094, 2096	\labelformat 3
H	\language_name 15
\hyperlink 45	\LoadDictionaryFor 16, 473, 477, 525, 529, 542, 546, 552
I	M
\IfBooleanTF 848	\MessageBreak 10
\IfFormatAtLeastTF 3, 4	\meta 1172
\IfTranslation 142	msg commands:
int commands:	\msg_line_context: 102, 108, 115, 129, 133, 135, 137, 139
\int_case:nnTF 1375, 1401, 1432, 1596, 1693, 1729	\msg_new:nnn ... 100, 106, 111, 113, 118, 123, 125, 127, 132, 134, 136, 138
\int_compare:nNnTF 973, 1028, 1095, 1111, 1141, 1143, 1188, 1343, 1388, 1422, 1585, 1587, 1650, 1674, 1716, 2177, 2183, 2203, 2209	\msg_warning:nn 260, 285, 389, 395, 568, 589
\int_compare_p:nNn 1157, 1165, 1836, 1847, 1955	\msg_warning:nnn 228, 660, 747, 802, 1336, 1493, 1879, 1926
\int_eval:n 90	\msg_warning:nnnn 174, 1036, 1059, 1082, 1120
\int_incr:N 1623, 1663, 1665, 1679, 1681, 1685, 1687, 1776	N
\int_new:N 852, 853, 1202, 1203, 1204, 1205	\newcounter 4
\int_set:Nn ... 1142, 1144, 1148, 1151	\NewDocumentCommand 155, 157, 635, 641, 709, 812, 846
\int_use:N 33, 35, 50	\NewHook 448
	\newtheorem 55
	\nobreakspace 2222, 2239, 2240, 2242, 2243, 2245, 2247, 2402, 2403, 2405, 2406, 2408, 2410, 2551, 2552, 2554, 2555, 2557, 2559, 2696, 2697, 2699, 2700, 2702, 2704, 2855, 2856, 2858, 2859, 2861, 2863
	\noexpand 121
	P
	\PackageError 7
	\pagenumbering 6

<code>\zref@refused</code>	1330	999, 1001, 1010, 1012, 1050, 1073,
<code>\zref@wrapper@babel</code>	24, 813	1100, 1102, 1104, 1106, 1268, 1269,
<code>\textendash</code>	2230	1272, 1277, 1381, 1382, 1505, 1536,
<code>\texttt</code>	832	1654, 1655, 1678, 1838, 1839, 1851
<code>\the</code>	3	<code>\tl_set_eq:NN</code>
<code>\thechapter</code>	55	459, 465, 497, 503, 515, 519, 536, 1616
<code>\thepage</code>	6, 95	<code>\tl_tail:N</code> 1101, 1103, 1105, 1107
<code>\thesection</code>	55	<code>\l_tmpa_tl</code>
tl commands:		1006,
<code>\c_empty_tl</code>	909, 924,	1010, 1019, 1047, 1050, 1053, 1093
	926, 988, 992, 996, 1000, 1275, 1280	<code>\l_tmpb_tl</code>
<code>\c_novalue_tl</code>	621, 692	1007,
<code>\tl_clear:N</code> ..	712, 720, 1006, 1007,	1012, 1020, 1070, 1073, 1076, 1093
	1235, 1236, 1237, 1238, 1239, 1261,	
	1618, 1619, 1620, 1621, 1662, 1827,	
	1830, 1854, 1878, 1925, 2164, 2166	
<code>\tl_gset:Nn</code>	95	
<code>\tl_head:N</code>		
	.. 1011, 1013, 1096, 1098, 1112, 1114	
<code>\tl_if_empty:NTF</code>	71, 741, 771,	
	800, 910, 1334, 1488, 1841, 1856, 2030	
<code>\tl_if_empty:nTF</code>		
 163, 719, 1440, 1455,	
	1470, 1711, 1736, 1747, 1760, 1829	
<code>\tl_if_empty_p:N</code> ...	932, 933, 941,	
	942, 949, 950, 1305, 1306, 1313,	
	1314, 1939, 1949, 1953, 2112, 2156	
<code>\tl_if_empty_p:n</code>	1019, 1020, 1047, 1070	
<code>\tl_if_eq:NNTF</code>	956, 1093, 1317	
<code>\tl_if_eq:NnTF</code>	866,	
	899, 1147, 1150, 1176, 1179, 1266, 2171	
<code>\tl_if_eq:nnTF</code>		
	... 969, 1023, 1139, 2173, 2195, 2199	
<code>\tl_if_in:NnTF</code>	1052, 1075	
<code>\tl_if_novalue:nTF</code>	624, 695	
<code>\tl_map_break:n</code>	81	
<code>\tl_map_tokens:Nn</code>	73	
<code>\tl_new:N</code>	89, 245, 445,	
	446, 447, 557, 593, 639, 640, 854,	
	855, 856, 857, 858, 859, 860, 861,	
	1198, 1199, 1200, 1201, 1206, 1210,	
	1211, 1212, 1214, 1215, 1216, 1217,	
	1218, 1220, 1221, 1222, 1223, 1224,	
	1225, 1226, 1227, 1228, 1230, 1231	
<code>\tl_put_left:Nn</code>	1522, 1529, 1578	
<code>\tl_put_right:Nn</code>	1390, 1406,	
	1415, 1442, 1452, 1467, 1698, 1709,	
	1734, 1745, 1758, 1842, 1843, 1852	
<code>\tl_reverse_items:n</code>		
 920, 920, 990, 994, 998, 1002	
<code>\tl_set:Nn</code>		
	. 250, 252, 258, 261, 277, 286, 470,	
	507, 551, 645, 711, 724, 908, 923,	
	925, 987, 989, 991, 993, 995, 997,	

U

use commands:

<code>\use:N</code>	21
<code>\UseHook</code>	555

Z

<code>\zcDeclareTranslations</code> .	20, 22, 103, 709
<code>\zcDicDefaultTransl</code>	8, 155, 2239,
	2240, 2241, 2242, 2243, 2244, 2245,
	2246, 2247, 2248, 2249, 2250, 2251,
	2402, 2403, 2404, 2405, 2406, 2407,
	2408, 2409, 2410, 2551, 2552, 2553,
	2554, 2555, 2556, 2557, 2558, 2559,
	2696, 2697, 2698, 2699, 2700, 2701,
	2702, 2703, 2704, 2855, 2856, 2857,
	2858, 2859, 2860, 2861, 2862, 2863
<code>\zcDicTypeTransl</code>	
	.. 8, 155, 2253, 2254, 2255, 2256,
	2258, 2259, 2260, 2261, 2263, 2264,
	2265, 2266, 2268, 2269, 2270, 2271,
	2272, 2273, 2274, 2275, 2277, 2278,
	2279, 2280, 2282, 2283, 2284, 2285,
	2286, 2287, 2289, 2290, 2291, 2292,
	2294, 2295, 2296, 2297, 2298, 2299,
	2300, 2301, 2303, 2304, 2305, 2306,
	2308, 2309, 2310, 2311, 2313, 2314,
	2315, 2316, 2318, 2319, 2320, 2321,
	2323, 2324, 2325, 2326, 2327, 2328,
	2329, 2330, 2331, 2332, 2334, 2335,
	2336, 2337, 2339, 2340, 2341, 2342,
	2344, 2345, 2346, 2347, 2349, 2350,
	2351, 2352, 2354, 2355, 2356, 2357,
	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, 2412, 2413, 2414, 2415,
	2417, 2418, 2419, 2420, 2422, 2423,
	2424, 2425, 2427, 2428, 2429, 2430,
	2432, 2433, 2434, 2435, 2437, 2438,
	2439, 2440, 2442, 2443, 2444, 2445,

2447, 2448, 2449, 2450, 2451, 2452,	2885, 2886, 2887, 2888, 2890, 2891,
2453, 2454, 2456, 2457, 2458, 2459,	2892, 2893, 2895, 2896, 2897, 2898,
2461, 2462, 2463, 2464, 2466, 2467,	2900, 2901, 2902, 2903, 2905, 2906,
2468, 2469, 2471, 2472, 2473, 2474,	2907, 2908, 2910, 2911, 2912, 2913,
2476, 2477, 2478, 2479, 2480, 2481,	2915, 2916, 2917, 2918, 2920, 2921,
2483, 2484, 2485, 2486, 2488, 2489,	2922, 2923, 2925, 2926, 2927, 2928,
2490, 2491, 2493, 2494, 2495, 2496,	2929, 2930, 2932, 2933, 2934, 2935,
2498, 2499, 2500, 2501, 2503, 2504,	2937, 2938, 2939, 2940, 2942, 2943,
2505, 2506, 2508, 2509, 2510, 2511,	2944, 2945, 2947, 2948, 2949, 2950,
2513, 2514, 2515, 2516, 2518, 2519,	2952, 2953, 2954, 2955, 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, 2561, 2562,	2982, 2983, 2984, 2985, 2987, 2988,
2563, 2564, 2566, 2567, 2568, 2569,	2989, 2990, 2992, 2993, 2994, 2995
2571, 2572, 2573, 2574, 2576, 2577,	\zcheck 121
2578, 2579, 2581, 2582, 2583, 2584,	\zcpageref 25, 846
2586, 2587, 2588, 2589, 2591, 2592,	\zceref 24, 26, 33, 35, 812, 849, 850
2593, 2594, 2596, 2597, 2598, 2599,	\zcRefTypeSetup 20, 21, 109, 641
2601, 2602, 2603, 2604, 2606, 2607,	\zcsetup 20, 635
2608, 2609, 2611, 2612, 2613, 2614,	zrefcheck commands:
2616, 2617, 2618, 2619, 2621, 2622,	\zrefcheck_zceref_beg_label: ... 826
2623, 2624, 2625, 2626, 2628, 2629,	\zrefcheck_zceref_end_label_-
2630, 2631, 2633, 2634, 2635, 2636,	maybe: 840
2638, 2639, 2640, 2641, 2643, 2644,	\zrefcheck_zceref_run_checks_on_-
2645, 2646, 2648, 2649, 2650, 2651,	labels:n 841
2653, 2654, 2655, 2656, 2658, 2659,	zrefclever internal commands:
2660, 2661, 2663, 2664, 2665, 2666,	\l__zrefclever_abbrev_bool 431, 435, 1845
2668, 2669, 2670, 2671, 2673, 2674,	\l__zrefclever_capitalize_bool .. 413, 417, 1833
2675, 2676, 2678, 2679, 2680, 2681,	\l__zrefclever_capitalize_first_-
2683, 2684, 2685, 2686, 2688, 2689,	bool 414, 423, 1835
2690, 2691, 2706, 2707, 2708, 2709,	__zrefclever_counter_reset_by:n
2711, 2712, 2713, 2714, 2716, 2717, 5, 10, 39, 41, 43, 48, 50, 52, 57
2718, 2719, 2721, 2722, 2723, 2724,	__zrefclever_counter_reset_by_-
2725, 2726, 2727, 2728, 2730, 2731,	aux:nn 64, 67
2732, 2733, 2735, 2736, 2737, 2738,	__zrefclever_counter_reset_by_-
2739, 2740, 2742, 2743, 2744, 2745,	auxi:nnn 74, 78
2747, 2748, 2749, 2750, 2751, 2752,	\l__zrefclever_counter_resetby_-
2753, 2754, 2756, 2757, 2758, 2759,	prop 4, 10, 60, 61, 221, 233
2761, 2762, 2763, 2764, 2766, 2767,	\l__zrefclever_counter_resetters_-
2768, 2769, 2771, 2772, 2773, 2774,	seq 4, 10, 63, 195, 202, 205
2776, 2777, 2778, 2779, 2780, 2781,	\l__zrefclever_counter_type_prop
2782, 2783, 2784, 2785, 2787, 2788, 3, 9, 25, 28, 167, 179
2789, 2790, 2792, 2793, 2794, 2795,	\l__zrefclever_current_language_-
2797, 2798, 2799, 2800, 2802, 2803,	tl 15, 16, 447, 466, 504, 515
2804, 2805, 2807, 2808, 2809, 2810,	__zrefclever_declare_default_-
2812, 2813, 2814, 2815, 2817, 2818,	transl:nnn 8, 153, 2222,
2819, 2820, 2822, 2823, 2824, 2825,	2223, 2224, 2225, 2226, 2227, 2228,
2827, 2828, 2829, 2830, 2832, 2833,	2229, 2230, 2231, 2232, 2233, 2234
2834, 2835, 2837, 2838, 2839, 2840,	__zrefclever_declare_transl:nnn
2842, 2843, 2844, 2845, 2847, 2848, 8, 150, 743, 773, 777, 806
2849, 2850, 2865, 2866, 2867, 2868,	
2870, 2871, 2872, 2873, 2875, 2876,	
2877, 2878, 2880, 2881, 2882, 2883,	

```

\__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 ... 1345, 1346, 1347, 2149
\__zrefclever_get_option_with-
  transl:nN ..... 55, 1246, 1247,
1248, 1249, 1348, 1349, 1350, 1351,
1352, 1353, 1354, 1355, 1356, 2103
\__zrefclever_get_ref:n 1393, 1409,
1421, 1425, 1445, 1458, 1461, 1473,
1476, 1510, 1530, 1701, 1714, 1719,
1739, 1750, 1753, 1763, 1766, 1778
\__zrefclever_get_ref_first: ...
..... 35, 45, 1523, 1579, 1962
\__zrefclever_get_transl:nnn ...
. 8, 147, 1870, 1902, 1917, 2132, 2142
\__zrefclever_if_transl:nn . 140, 146
\__zrefclever_if_transl:nnTF ...
..... 7, 140, 1863, 1895, 1910, 2128
\l__zrefclever_label_a_tl .....
.... 854, 1258, 1275, 1291, 1330,
1331, 1337, 1381, 1393, 1409, 1425,
1461, 1476, 1503, 1510, 1641, 1645,
1654, 1678, 1701, 1719, 1753, 1766
\l__zrefclever_label_b_tl .....
854, 1261, 1264, 1280, 1293, 1298, 1645
\l__zrefclever_label_count_int ..
..... 34, 1202,
1240, 1343, 1375, 1622, 1650, 1776
\l__zrefclever_label_enclcnt_a-
  tl ..... 854,
987, 989, 990, 1011, 1076, 1100, 1101
\l__zrefclever_label_enclcnt_b-
  tl ..... 854,
991, 993, 994, 1013, 1053, 1102, 1103
\l__zrefclever_label_enclval_a-
  tl ..... 854,
995, 997, 998, 1096, 1104, 1105, 1112
\l__zrefclever_label_enclval_b-
  tl ..... 854,
999, 1001, 1002, 1098, 1106, 1107, 1114
\l__zrefclever_label_type_a_tl ..
..... 854, 908, 910, 912,
915, 923, 932, 941, 949, 956, 1147,
1176, 1268, 1272, 1305, 1313, 1318,
1334, 1382, 1655, 2112, 2115, 2119,
2124, 2130, 2134, 2156, 2159, 2163
\l__zrefclever_label_type_b_tl ..
854, 925, 933, 942, 950, 956, 1150,
1179, 1269, 1277, 1306, 1314, 1318
\__zrefclever_label_type_put_-
  new_right:n ..... 27, 869, 906

\l__zrefclever_label_types_seq ..
..... 27, 862, 865, 912, 915, 1174
\__zrefclever_labels_in_sequence:nn
..... 1502, 1644, 2169
\l__zrefclever_last_of_type_bool
..... 33, 1195, 1289, 1294, 1295,
1299, 1308, 1319, 1320, 1323, 1361
\l__zrefclever_lastsep_tl . 1218,
1352, 1408, 1424, 1444, 1460, 1472
\l__zrefclever_link_star_bool ...
..... 814, 821, 1783, 1938, 2048
\l__zrefclever_listsep_tl .....
... 1217, 1351, 1420, 1457, 1700,
1713, 1718, 1738, 1749, 1752, 1762
\l__zrefclever_main_language_tl .
.. 15, 16, 446, 460, 498, 519, 536, 551
\l__zrefclever_name_format_-
  fallback_tl ..... 1228,
1851, 1854, 1856, 1892, 1914, 1921
\l__zrefclever_name_format_tl ...
..... 1228,
1838, 1839, 1842, 1843, 1851, 1852,
1860, 1867, 1874, 1887, 1899, 1906
\l__zrefclever_name_in_link_bool
.. 1228, 1538, 1942, 1958, 1959, 1967
\l__zrefclever_namefont_tl 1210,
1345, 1541, 1567, 1992, 2022, 2037
\l__zrefclever_nameinlink_str ...
..... 398, 403,
405, 407, 409, 1940, 1946, 1948, 1952
\l__zrefclever_namesep_tl .....
.. 1214, 1348, 1995, 2025, 2033, 2040
\l__zrefclever_next_is_same_bool
..... 34, 53, 1204,
1634, 1664, 1680, 1686, 2189, 2215
\l__zrefclever_next_maybe_range_-
  bool .....
.. 34, 53, 1204, 1498, 1507, 1633,
1660, 1671, 2181, 2188, 2207, 2214
\l__zrefclever_noabbrev_first_-
  bool ..... 432, 441, 1848
\l__zrefclever_noteseptl .....
..... 833, 1223, 1249
\__zrefclever_page_format_aux: ..
..... 90, 94
\g__zrefclever_page_format_tl ...
..... 6, 89, 95, 98
\l__zrefclever_pairsep_tl .....
..... 1216, 1350, 1392, 1508
\__zrefclever_prop_put_non_-
  empty:Nnn ..... 9, 161, 178, 232
\l__zrefclever_range_beg_label_-
  tl ..... 34, 1204, 1239,
1421, 1440, 1445, 1455, 1458, 1470,

```

1473, 1621, 1662, 1678, 1711, 1714,
 1736, 1739, 1747, 1750, 1760, 1763
 \l_zrefclever_range_count_int ..
 34,
 1204, 1242, 1401, 1433, 1624, 1663,
 1675, 1679, 1685, 1693, 1730, 1771
 \l_zrefclever_range_inhibit_
 next_bool 33, 34, 1204, 1639
 \l_zrefclever_range_same_count_
 int 34,
 1204, 1243, 1388, 1422, 1433, 1625,
 1665, 1681, 1687, 1716, 1730, 1772
 \l_zrefclever_rangeseq_tl
 1215, 1349, 1475, 1509, 1765
 \l_zrefclever_ref_language_tl ..
 15, 445, 459, 465, 470,
 497, 503, 507, 1864, 1871, 1896,
 1903, 1911, 1918, 2129, 2133, 2143
 \l_zrefclever_ref_options_prop ..
 21, 596, 625, 626, 2107, 2152
 \l_zrefclever_ref_property_tl ..
 11,
 245, 250, 252, 258, 261, 277, 286,
 866, 899, 1266, 1780, 1800, 1814,
 1970, 2003, 2043, 2077, 2092, 2171
 \l_zrefclever_ref_typeset_font_
 tl 593, 595, 1252
 \l_zrefclever_reffont_in_tl 1212,
 1347, 1789, 1812, 2000, 2055, 2089
 \l_zrefclever_reffont_out_tl ...
 1211, 1346,
 1786, 1809, 1997, 2016, 2052, 2086
 \l_zrefclever_refpos_in_tl 1227,
 1356, 1801, 1815, 2004, 2078, 2093
 \l_zrefclever_refpos_out_tl 1225,
 1354, 1804, 1817, 2017, 2081, 2095
 \l_zrefclever_refpre_in_tl 1226,
 1355, 1799, 1813, 2001, 2075, 2090
 \l_zrefclever_refpre_out_tl 1224,
 1353, 1787, 1810, 1998, 2053, 2087
 \l_zrefclever_setup_language_tl
 639, 711, 743, 773, 777, 806
 \l_zrefclever_setup_type_tl ...
 639, 645, 698, 703,
 712, 720, 724, 741, 771, 778, 800, 807
 \l_zrefclever_sort_decided_bool
 919, 1004, 1008, 1027, 1038,
 1055, 1061, 1078, 1084, 1110, 1122
 _zrefclever_sort_default:nn ...
 26–28, 901, 921
 _zrefclever_sort_default_
 different_types:nn .. 13, 963, 1130
 _zrefclever_sort_default_same_
 type:nn 959, 985
 _zrefclever_sort_labels:
 27, 33, 830, 863
 _zrefclever_sort_page:nn
 33, 900, 1186
 \l_zrefclever_sort_prior_a_int ..
 852, 1132, 1141, 1142, 1148, 1158, 1166
 \l_zrefclever_sort_prior_b_int ..
 853, 1133, 1143, 1144, 1151, 1159, 1167
 \l_zrefclever_tlastsep_tl
 1222, 1248, 1610
 \l_zrefclever_tlistsep_tl
 1221, 1247, 1588
 \l_zrefclever_tpairsep_tl
 1220, 1246, 1604
 \l_zrefclever_type<type>_
 options_prop 21
 \l_zrefclever_type_count_int ...
 34, 1202, 1241, 1585,
 1587, 1596, 1623, 1836, 1847, 1955
 \l_zrefclever_type_first_label_
 tl 1197, 1237, 1381, 1490,
 1499, 1503, 1530, 1546, 1549, 1554,
 1560, 1619, 1654, 1826, 1964, 1970,
 1976, 1978, 1982, 1987, 2002, 2043,
 2060, 2062, 2066, 2071, 2076, 2091
 \l_zrefclever_type_first_label_
 type_tl 1197,
 1238, 1382, 1494, 1620, 1655, 1829,
 1859, 1866, 1873, 1880, 1886,
 1891, 1898, 1905, 1913, 1920, 1927
 _zrefclever_type_name_setup: ..
 35, 1518, 1824
 \l_zrefclever_type_name_tl .. 46,
 1228, 1562, 1568, 1827, 1830, 1861,
 1870, 1878, 1888, 1893, 1902, 1917,
 1925, 1939, 1993, 2023, 2030, 2038
 \l_zrefclever_typeset_compress_
 bool 341, 344, 1636
 \l_zrefclever_typeset_labels_
 seq ... 1197, 1234, 1258, 1259, 1264
 \l_zrefclever_typeset_last_bool
 33, 1195,
 1255, 1256, 1262, 1288, 1593, 1954
 \l_zrefclever_typeset_name_bool
 292, 299, 304, 309, 1520, 1534
 \l_zrefclever_typeset_queue_
 curr_tl 1197, 1236, 1390,
 1406, 1415, 1442, 1452, 1467, 1488,
 1505, 1522, 1529, 1536, 1579, 1600,
 1605, 1611, 1617, 1618, 1698, 1709,
 1734, 1745, 1758, 1841, 1949, 1953
 \l_zrefclever_typeset_queue_
 prev_tl 1197, 1235, 1589, 1617

\l_zrefclever_typeset_range_- 357, 364,
bool 350, 353, 829, 1486	369, 374, 384, 390, 1783, 1937, 2047
\l_zrefclever_typeset_ref_bool .	\l_zrefclever_warn_hyperref_-
..... 291, 298, 303, 308, 1520, 1527	bool 358, 365, 370, 375, 388
_zrefclever_typeset_refs:	_zrefclever_zcref:nnn 813, 816
..... 33, 34, 45, 46, 49, 831, 1232	_zrefclever_zcref:nnnn . 24, 26, 816
_zrefclever_typeset_refs_aux_-	\l_zrefclever_zcref_labels_seq .
last_of_type: 1364, 1372 26, 814, 820, 842, 869, 871, 1234
_zrefclever_typeset_refs_aux_-	\l_zrefclever_zcref_note_tl ...
not_last_of_type: 1368, 1628 557, 560, 834
\l_zrefclever_typeset_sort_bool	\l_zrefclever_zcref_with_check_-
..... 317, 320, 828	bool 564, 579, 825, 838
\l_zrefclever_typesort_seq	\l_zrefclever_zrefcheck_-
..... 13, 326, 331, 332, 338, 1137	available_bool 563, 574, 585, 824, 837
\l_zrefclever_use_hyperref_bool	