

The zref-clever package implementation*

Gustavo Barros[†]

2021-09-13

Contents

| | | |
|-----------|----------------------------------|-----------|
| 1 | Initial setup | 1 |
| 2 | Dependencies | 2 |
| 3 | zref setup | 2 |
| 4 | Plumbing | 6 |
| 4.1 | Messages | 6 |
| 4.2 | Translations aux | 7 |
| 4.3 | Options | 7 |
| 5 | Type format | 18 |
| 5.1 | \zcRefTypeSetup | 18 |
| 5.2 | \zcDeclareTranslations | 20 |
| 6 | \zceref | 22 |
| 7 | \zcpageref | 23 |
| 8 | Sorting | 23 |
| 9 | Typesetting | 31 |
| 10 | Translations | 52 |
| | Index | 67 |

*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 ‘ltxcounts.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 Section ??). Apparently, this relation might just be stored in a property list, rather than in the label itself. However, there are cases in which we must 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   { \use:c { @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 }

```

TODO Stopped here.

The need to check this [nested counters] has some implications to the data we store in the label. Since we cannot do this verification when we set up the *reference type*, because at this point we could only check existing counters, and they may be defined “later” or “never”, the counter reset chain must be stored (names and values) with the label itself (this is done in properties `zc@enclcnt` and `zc@enclval`).

i) the counter *value*, as a number; ii) the counter (and value) of the set of counters which may trigger a reset of the current counter.

The second one is trickier. For starters, the counters which may reset the current counter are not retrievable from the counter itself, because this information is stored with the counter that does the resetting, not with the one that gets reset (the list is stored in `\cl@<counter>` with format `\@elt{countera}\@elt{counterb}\@elt{counterc}`, again

see section ‘ltcounts.dtx’ in ‘source2e’). Besides, there may be a chain of resetting counters, which must be taken into account. The procedure below examines a set of counters, those included in `\l__zrefclever_counter_resettters_seq`, and for each counter retrieves its “enclosing counters” recursively. 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.

Recursively generate a *sequence* of “enclosing counters” and values, for a given $\{\langle counter \rangle\}$ 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.

```

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:N \cs:w c@ \__zrefclever_counter_reset_by:n {#1} \cs_end: }
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`.)

Auxiliary functions 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. In particular `__zrefclever_counter_reset_by:n` leaves in the stream the “enclosing counter” which resets $\{\langle counter \rangle\}$.

```

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 { cl@ #2 }
72       {
73         \tl_map_tokens:cn { cl@ #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`, `__zrefclever_counter_reset_by_aux:nn`, and `__zrefclever_counter_reset_by_auxi:nnn`.)

Finally, add `zc@enclcnt` and `zc@enclval` to `zref`'s 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, the “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 can be used for that. But we can decide whether two labels can be compressed or not based on this format: if they are identical, we can compress them, otherwise, we can’t. `cleveref` actually resets the counter to “1” with `\setcounter`, which is a global operation, and restores it in sequence. Here we adopt a more cautious approach of locally redefining `\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}`.

`__zrefclever_page_numbering:`

```

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 }

```

(End definition for `__zrefclever_page_numbering:.`)

Another property which we don't need to handle at the data provision side, but need to cater for in the retrieval side, are the `url` / `urluse` properties from the `zref-xr` module, which are 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 } { countertype-requires-value }
112 { The~'countertype'~key~'#1'~requires~a~value. }
113 \msg_new:nnn { zref-clever } { counterresetby-requires-value }
114 { The~'counterresetby'~key~'#1'~requires~a~value. }
115 \msg_new:nnn { zref-clever } { missing-zref-titleref }
116 {
117   Option~'ref=title'~requested~\msg_line_context:~
118   But~package~'zref-titleref'~is~not~loaded,~falling~back~to~default~'ref'.
119 }
120 \msg_new:nnn { zref-clever } { hyperref-preamble-only }
121 {
122   Option~'hyperref'~only~available~in~the~preamble. \iow_newline:
123   Use~the~starred~version~of~'\noexpand\zcheck'~instead.
124 }
125 \msg_new:nnn { zref-clever } { missing-hyperref }

```

```

126 { Missing~'hyperref'~package. \iow_newline: Setting~'hyperref=false'. }
127 \msg_new:nnn { zref-check } { check-document-only }
128 { Option~'check'~only~available~in~the~document. }
129 \msg_new:nnn { zref-clever } { missing-zref-check }
130 {
131   Option~'check'~requested~\msg_line_context:..
132   But~package~'zref-check'~is~not~loaded,~can't~run~the~checks.
133 }
134 \msg_new:nnn { zref-clever } { counters-not-nested }
135 { Counters~not~nested~for~labels~'#1'~and~'#2'~\msg_line_context:.. }
136 \msg_new:nnn { zref-clever } { missing-type }
137 { Reference~type~undefined~for~label~'#1'~\msg_line_context:.. }
138 \msg_new:nnn { zref-clever } { missing-name }
139 { Name~undefined~for~type~'#1'~\msg_line_context:.. }
140 \msg_new:nnn { zref-clever } { single-element-range }
141 { Range~for~type~'#1'~resulted~in~single~element~\msg_line_context:.. }

```

4.2 Translations aux

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

```

142 \prg_new_conditional:Npnn \__zrefclever_if_translation:nn #1#2 { p , TF }
143 {
144   \IfTranslation {#1} {#2}
145   { \prg_return_true: }
146   { \prg_return_false: }
147 }
148 \prg_generate_conditional_variant:Nnn \__zrefclever_if_translation:nn { xx } { p , TF }
149 \cs_new_protected:Npn \__zrefclever_get_translation_for:nnn #1#2#3
150 { \SaveTranslationFor{#1}{#2}{#3} }
151 \cs_generate_variant:Nn \__zrefclever_get_translation_for:nnn { nxx }
152 \cs_new_protected:Npn \__zrefclever_declare_translation:nnn #1#2#3
153 { \declaretranslation {#1} {#2} {#3} }
154 \cs_generate_variant:Nn \__zrefclever_declare_translation:nnn { xxn , xxx }
155
156 % <lang><key><transl>
157 \cs_new_protected:Npn \__zrefclever_add_default_translation:nnn #1#2#3
158 { \addtranslation {#1} { zrefclever-default- #2 } {#3} }
159
160 % <lang><type><key><transl>
161 \cs_new_protected:Npn \__zrefclever_add_type_translation:nnnn #1#2#3#4
162 { \addtranslation {#1} { zrefclever-type- #2 - #3 } {#4} }

```

Functions for use in dictionary files. The dictionary file commands cannot rely on expl3 syntax, so we define “document” ones.

```

163 % <key><transl>
164 \NewDocumentCommand \zcDicDefaultTransl { m m }
165 { \ProvideDictTranslation { zrefclever-default- #1 } {#2} }
166 % <type><key><transl>
167 \NewDocumentCommand \zcDicTypeTransl { m m m }
168 { \ProvideDictTranslation { zrefclever-type- #1 - #2 } {#3} }

```

4.3 Options

countertype option

`\l_zrefclever_counter_type_prop`

Variable storing a mapping from “counter” to “reference type”.

```
169 \prop_new:N \l__zrefclever_counter_type_prop
```

(End definition for `\l__zrefclever_counter_type_prop`.)

```
170 \cs_new_protected:Npn \__zrefclever_prop_put_non_empty:Nnn #1#2#3
171 {
172   \tl_if_empty:nTF {#3}
173     { \prop_remove:Nn #1 {#2} }
174     { \prop_put:Nnn #1 {#2} {#3} }
175 }
176 \keys_define:nn { zref-clever }
177 {
178   countertype .code:n =
179   {
180     \keyval_parse:nnn
181     { \msg_warning:nnn { zref-clever } { countertype-requires-value } }
182     { \__zrefclever_prop_put_non_empty:Nnn \l__zrefclever_counter_type_prop }
183     {#1}
184   } ,
185   countertype .value_required:n = true ,
186   countertype .initial:n =
187   {
188     subsection      = section ,
189     subsubsection    = section ,
190     subparagraph     = paragraph ,
191     enumi            = item ,
192     enumii           = item ,
193     enumiii          = item ,
194     enumiv           = item ,
195   } ,
196 }
```

counterresetters option

`\l_zrefclever_counter_resetters_seq`

Stores the list of counters which are potential “enclosing counters” for other counters.

```
197 \seq_new:N \l__zrefclever_counter_resetters_seq
```

(End definition for `\l__zrefclever_counter_resetters_seq`.)

```
198 \keys_define:nn { zref-clever }
199 {
200   counterresetters .code:n =
201   {
202     \clist_map_inline:nn {#1}
203     {
204       \seq_if_in:NnF \l__zrefclever_counter_resetters_seq {##1}
205       { \seq_put_right:Nn \l__zrefclever_counter_resetters_seq {##1} }
206     }
207   } ,
208   counterresetters .initial:n =
209   {
```



```

210     part ,
211     chapter ,
212     section ,
213     subsection ,
214     subsubsection ,
215     paragraph ,
216     subparagraph ,
217   },
218   typesort .value_required:n = true ,
219 }

```

counterresetby option

`\l__zrefclever_counter_resetby_prop` Variable storing a mapping from “counter” to the counter which resets it.

```

220 \prop_new:N \l__zrefclever_counter_resetby_prop

```

(End definition for `\l__zrefclever_counter_resetby_prop`.)

```

221 \keys_define:nn { zref-clever }
222 {
223   counterresetby .code:n =
224   {
225     \keyval_parse:nnn
226     { \msg_warning:nnn { zref-clever } { counterresetby-requires-value } }
227     { \l__zrefclever_prop_put_non_empty:Nnn \l__zrefclever_counter_resetby_prop }
228     {#1}
229   } ,
230   counterresetby .value_required:n = true ,
231   counterresetby .initial:n =
232   {

```

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. TODO This list should probably be extended for ‘enumitem’, conditioned on it being loaded.

```

233     enumii = enumi ,
234     enumiii = enumii ,
235     enumiv = enumiii ,
236   } ,
237 }

```

ref option

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

```

238 \tl_new:N \l__zrefclever_ref_property_tl
239 \bool_new:N \l__zrefclever_page_ref_bool
240 \keys_define:nn { zref-clever }
241 {
242   ref .choice: ,
243   ref / zc@thecnt .code:n =
244   {
245     \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
246     \bool_set_false:N \l__zrefclever_page_ref_bool
247   } ,

```

```

248   ref / page .code:n =
249   {
250     \tl_set:Nn \l__zrefclever_ref_property_tl { page }
251     \bool_set_true:N \l__zrefclever_page_ref_bool
252   } ,
253   ref / title .code:n =
254   {
255     \AddToHook { begindocument }
256     {
257       \@ifpackageloaded { zref-titleref }
258       {
259         \tl_set:Nn \l__zrefclever_ref_property_tl { title }
260         \bool_set_false:N \l__zrefclever_page_ref_bool
261       }
262       {
263         \msg_warning:nn { zref-clever } { missing-zref-titleref }
264         \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
265         \bool_set_false:N \l__zrefclever_page_ref_bool
266       }
267     }
268   } ,
269   ref .initial:n = zc@thecnt ,
270   ref .value_required:n = true ,
271   page .meta:n = { ref = page } ,
272   page .value_forbidden:n = true ,
273 }
274
275 \AddToHook { begindocument }
276 {
277   \@ifpackageloaded { zref-titleref }
278   {
279     \keys_define:nn { zref-clever }
280     {
281       ref / title .code:n =
282       {
283         \tl_set:Nn \l__zrefclever_ref_property_tl { title }
284         \bool_set_false:N \l__zrefclever_page_ref_bool
285       }
286     }
287   }
288   {
289     \keys_define:nn { zref-clever }
290     {
291       ref / title .code:n =
292       {
293         \msg_warning:nn { zref-clever } { missing-zref-titleref }
294         \tl_set:Nn \l__zrefclever_ref_property_tl { zc@thecnt }
295         \bool_set_false:N \l__zrefclever_page_ref_bool
296       }
297     }
298   }
299 }

```

Currently, we restrict ‘ref=’ to these two (or three) alternatives, but there might be a case for making this more flexible. The infrastructure can already handle receiving

an arbitrary property, as long as one is satisfied with sorting and compressing from the default counter. If more flexibility is granted, one thing *must* be handled at this point: the existence of the property itself, as far as `zref` is concerned. This because typesetting relies on the check `\zref@ifrefcontainsprop`, which *presumes* the property is defined and silently expands the *true* branch if it is not (see <https://github.com/ho-tex/zref/issues/13>, thanks Ulrike Fischer). Therefore, before adding anything to `\l__zrefclever_ref_property_tl`, check if first here with `\zref@ifpropundefined`: close it at the door.

typeset option

```

300 \bool_new:N \l__zrefclever_typeset_ref_bool
301 \bool_new:N \l__zrefclever_typeset_name_bool
302 \keys_define:nn { zref-clever }
303 {
304     typeset .choice: ,
305     typeset / both .code:n =
306     {
307         \bool_set_true:N \l__zrefclever_typeset_ref_bool
308         \bool_set_true:N \l__zrefclever_typeset_name_bool
309     } ,
310     typeset / ref .code:n =
311     {
312         \bool_set_true:N \l__zrefclever_typeset_ref_bool
313         \bool_set_false:N \l__zrefclever_typeset_name_bool
314     } ,
315     typeset / name .code:n =
316     {
317         \bool_set_false:N \l__zrefclever_typeset_ref_bool
318         \bool_set_true:N \l__zrefclever_typeset_name_bool
319     } ,
320     typeset .initial:n = both ,
321     typeset .value_required:n = true ,
322
323     noname .meta:n = { typeset = ref } ,
324     noname .value_forbidden:n = true ,
325 }

```

sort option

User option, sort labels ranges or not

```

326 \bool_new:N \l__zrefclever_typeset_sort_bool
327 \keys_define:nn { zref-clever }
328 {
329     sort .bool_set:N = \l__zrefclever_typeset_sort_bool ,
330     sort .initial:n = true ,
331     sort .default:n = true ,
332     nosort .meta:n = { sort = false } ,
333     nosort .value_forbidden:n = true ,
334 }

```

typesort option

```

335 \seq_new:N \l__zrefclever_typesort_seq

```

```

336 \keys_define:nn { zref-clever }
337 {
338   typesort .code:n =
339   {
340     \seq_set_from_clist:Nn \l__zrefclever_typesort_seq {#1}
341     % Reverse the sequence, since the sort priorities are computed in the
342     % negative range, so that we can implicitly rely on '0' being the
343     % 'last value'.
344     \seq_reverse:N \l__zrefclever_typesort_seq
345   } ,
346   typesort .initial:n =
347   { part , chapter , section , paragraph } ,
348   typesort .value_required:n = true ,
349   notypesort .code:n =
350   { \seq_clear:N \l__zrefclever_typesort_seq } ,
351   notypesort .value_forbidden:n = true ,
352 }

```

comp option

User option, compress ranges or not

```

353 \bool_new:N \l__zrefclever_typeset_compress_bool
354 \keys_define:nn { zref-clever }
355 {
356   comp .bool_set:N = \l__zrefclever_typeset_compress_bool ,
357   comp .initial:n = true ,
358   comp .default:n = true ,
359   nocomp .meta:n = { comp = false } ,
360   nocomp .value_forbidden:n = true ,
361 }

```

range option

```

362 \bool_new:N \l__zrefclever_typeset_range_bool
363 \keys_define:nn { zref-clever }
364 {
365   range .bool_set:N = \l__zrefclever_typeset_range_bool ,
366   range .initial:n = false ,
367   range .default:n = true ,
368 }

```

hyperref option

\l__zrefclever_use_hyperref_bool
\l__zrefclever_warn_hyperref_bool

```

369 \bool_new:N \l__zrefclever_use_hyperref_bool
370 \bool_new:N \l__zrefclever_warn_hyperref_bool
371 \keys_define:nn { zref-clever }
372 {
373   hyperref .choice: ,
374   hyperref / auto .code:n =
375   {
376     \bool_set_true:N \l__zrefclever_use_hyperref_bool
377     \bool_set_false:N \l__zrefclever_warn_hyperref_bool
378   } ,
379   hyperref / true .code:n =

```

```

380     {
381       \bool_set_true:N \l__zrefclever_use_hyperref_bool
382       \bool_set_true:N \l__zrefclever_warn_hyperref_bool
383     } ,
384     hyperref / false .code:n =
385     {
386       \bool_set_false:N \l__zrefclever_use_hyperref_bool
387       \bool_set_false:N \l__zrefclever_warn_hyperref_bool
388     } ,
389     hyperref .initial:n = auto ,
390     hyperref .default:n = auto
391   }

```

(End definition for \l__zrefclever_use_hyperref_bool and \l__zrefclever_warn_hyperref_bool.)

```

392 \AddToHook { begindocument }
393 {
394   \@ifpackageloaded { hyperref }
395   {
396     \bool_if:NT \l__zrefclever_use_hyperref_bool
397     { \RequirePackage { zref-hyperref } }
398   }
399   {
400     \bool_if:NT \l__zrefclever_warn_hyperref_bool
401     { \msg_warning:nn { zref-clever } { missing-hyperref } }
402     \bool_set_false:N \l__zrefclever_use_hyperref_bool
403   }
404   \keys_define:nn { zref-clever }
405   {
406     hyperref .code:n =
407     { \msg_warning:nn { zref-clever } { hyperref-preamble-only } }
408   }
409 }

```

nameinlink option

\l__zrefclever_nameinlink_tl

```

410 \str_new:N \l__zrefclever_nameinlink_str
411 \keys_define:nn { zref-clever }
412 {
413   nameinlink .choice: ,
414   nameinlink / true .code:n =
415   { \str_set:Nn \l__zrefclever_nameinlink_str { true } } ,
416   nameinlink / false .code:n =
417   { \str_set:Nn \l__zrefclever_nameinlink_str { false } } ,
418   nameinlink / single .code:n =
419   { \str_set:Nn \l__zrefclever_nameinlink_str { single } } ,
420   nameinlink / tsingle .code:n =
421   { \str_set:Nn \l__zrefclever_nameinlink_str { tsingle } } ,
422   nameinlink .initial:n = tsingle ,
423   nameinlink .default:n = true ,
424 }

```

(End definition for \l__zrefclever_nameinlink_tl.)

cap capfirst options

```
425 \bool_new:N \l__zrefclever_capitalize_bool
426 \bool_new:N \l__zrefclever_capitalize_first_bool
427 \keys_define:nn { zref-clever }
428 {
429   cap .bool_set:N = \l__zrefclever_capitalize_bool ,
430   cap .initial:n = false ,
431   cap .default:n = true ,
432   nocap .meta:n = { cap = false },
433   nocap .value_forbidden:n = true ,
434
435   capfirst .bool_set:N = \l__zrefclever_capitalize_first_bool ,
436   capfirst .initial:n = false ,
437   capfirst .default:n = true ,
438
439   C .meta:n =
440     { capfirst = true , noabbrevfirst = true },
441   C .value_forbidden:n = true ,
442 }
```

abbrev noabbrevfirst option

```
443 \bool_new:N \l__zrefclever_abbrev_bool
444 \bool_new:N \l__zrefclever_noabbrev_first_bool
445 \keys_define:nn { zref-clever }
446 {
447   abbrev .bool_set:N = \l__zrefclever_abbrev_bool ,
448   abbrev .initial:n = false ,
449   abbrev .default:n = true ,
450   noabbrev .meta:n = { abbrev = false },
451   noabbrev .value_forbidden:n = true ,
452
453   noabbrevfirst .bool_set:N = \l__zrefclever_noabbrev_first_bool ,
454   noabbrevfirst .initial:n = false ,
455   noabbrevfirst .default:n = true ,
456 }
```

lang option

```
457 \tl_new:N \l__zrefclever_ref_language_tl
458 \tl_new:N \l__zrefclever_main_language_tl
459 \tl_new:N \l__zrefclever_current_language_tl
460 \NewHook { zref-clever / reflanguage }
461 \keys_define:nn { zref-clever }
462 {
463   lang .code:n =
464     {
465       \AddToHook { zref-clever / reflanguage }
466       {
467         \str_case:nnF {#1}
468         {
469           { main }
470           {
471             \tl_set_eq:NN
472             \l__zrefclever_ref_language_tl \l__zrefclever_main_language_tl

```

```

473         }
474
475         { current }
476         {
477             \tl_set_eq:NN
478             \l__zrefclever_ref_language_tl \l__zrefclever_current_language_tl
479         }
480     }
481     {
482         \tl_set:Nn \l__zrefclever_ref_language_tl {#1}
483         % If user specified a language at the preamble, make sure it
484         % is loaded.
485         \exp_args:Nx \file_if_exist:nTF
486         { zref-clever- \@trnslt@language {#1} .trsl }
487         { \LoadDictionaryFor {#1} { zref-clever } }
488         {
489             \exp_args:Nx \file_if_exist:nT
490             { zref-clever- \baselanguage {#1} .trsl }
491             { \LoadDictionaryFor {#1} { zref-clever } }
492         }
493     }
494 }
495 },
496 lang .initial:n = main ,
497 lang .value_required:n = true ,
498 }

```

\AtEndOfPackage so that it comes after \ProcessKeysOptions.

```

499 \AtEndOfPackage
500 {
501     \AddToHook { zref-clever / reflanguage }
502     {
503         \keys_define:nn { zref-clever }
504         {
505             lang .code:n =
506             {
507                 \str_case:nnF {#1}
508                 {
509                     { main }
510                     {
511                         \tl_set_eq:NN
512                         \l__zrefclever_ref_language_tl \l__zrefclever_main_language_tl
513                     }
514
515                     { current }
516                     {
517                         \tl_set_eq:NN
518                         \l__zrefclever_ref_language_tl \l__zrefclever_current_language_tl
519                     }
520                 }
521                 { \tl_set:Nn \l__zrefclever_ref_language_tl {#1} }
522             },
523             lang .value_required:n = true ,
524         }
525     }

```

```

526 }
    See https://tex.stackexchange.com/a/233178 (including Javier Bezos' comment).
    Also https://tex.stackexchange.com/a/281220 (including PLK's comments).
527 \AddToHook { begindocument / before }
528 {
529     % An internal alias for \pkg{translations}'s internal macro
530     % \cs{@trnslt@current@language}.
531     \tl_set_eq:NN \l__zrefclever_current_language_tl \@trnslt@current@language
532     % Getting main languages and, for each babel/polyglossia loaded language,
533     % load corresponding zref-clever dictionary.
534     \ifpackageloaded{babel}
535     {
536         \tl_set_eq:NN \l__zrefclever_main_language_tl \bbl@main@language
537         \clist_map_inline:Nn \bbl@loaded
538         {
539             % Funny enough, \pkg{translations} also loads its basic
540             % dictionaries for all languages loaded by babel or polyglossia.
541             % First, there is no way to disable this, even if we don't need
542             % them at all here. Second, \pkg{translations} sends messages of
543             % its own missing dictionaries to 'info' and everyone else's to
544             % 'warning'... So we have to control ourselves for missing
545             % dictionaries and load them only if available.
546             \exp_args:Nx \file_if_exist:nTF
547             { zref-clever- \@trnslt@language {#1} .trsl }
548             { \LoadDictionaryFor {#1} { zref-clever } }
549             {
550                 \exp_args:Nx \file_if_exist:nT
551                 { zref-clever- \baselanguage {#1} .trsl }
552                 { \LoadDictionaryFor {#1} { zref-clever } }
553             }
554         }
555     }
556     {
557         \ifpackageloaded{polyglossia}
558         {
559             \tl_set_eq:NN \l__zrefclever_main_language_tl \xpg@main@language
560             \clist_map_inline:Nn \xpg@loaded
561             {
562                 \exp_args:Nx \file_if_exist:nTF
563                 { zref-clever- \@trnslt@language {#1} .trsl }
564                 { \LoadDictionaryFor {#1} { zref-clever } }
565                 {
566                     \exp_args:Nx \file_if_exist:nT
567                     { zref-clever- \baselanguage {#1} .trsl }
568                     { \LoadDictionaryFor {#1} { zref-clever } }
569                 }
570             }
571         }
572         {
573             \tl_set:Nn \l__zrefclever_main_language_tl { english }
574             \LoadDictionaryFor { english } { zref-clever }
575         }
576     }
577     % *Then* we execute the package options stored in the 'reflanguage' hook.

```



```

578     \UseHook { zref-clever / reflanguage }
579   }

```

note option

```

580 \tl_new:N \l__zrefclever_zcref_note_tl
581 \keys_define:nn { zref-clever }
582 {
583   note .tl_set:N = \l__zrefclever_zcref_note_tl ,
584   note .value_required:n = true ,
585 }

```

check option

Integration with zref-check.

```

586 \bool_new:N \l__zrefclever_zrefcheck_available_bool
587 \bool_new:N \l__zrefclever_zcref_with_check_bool
588 \keys_define:nn { zref-clever }
589 {
590   check .code:n =
591     { \msg_warning:nn { zref-clever } { check-document-only } } ,
592 }
593 \AddToHook { begindocument }
594 {
595   \@ifpackageloaded { zref-check }
596   {
597     \bool_set_true:N \l__zrefclever_zrefcheck_available_bool
598     \keys_define:nn { zref-clever }
599     {
600       check .code:n =
601       {
602         \bool_set_true:N \l__zrefclever_zcref_with_check_bool
603         \keys_set:nn { zref-check / zcheck } {#1}
604       }
605     }
606   }
607   {
608     \bool_set_false:N \l__zrefclever_zrefcheck_available_bool
609     \keys_define:nn { zref-clever }
610     {
611       check .code:n =
612       { \msg_warning:nn { zref-clever } { missing-zref-check } }
613     }
614   }
615 }

```

Reference options

```

616 \tl_new:N \l__zrefclever_ref_typeset_font_tl
617 \keys_define:nn { zref-clever }
618 { font .tl_set:N = \l__zrefclever_ref_typeset_font_tl }

```

Only not necessarily type-specific options are pertinent here.

```

619 \prop_new:N \l__zrefclever_ref_options_prop
620 \clist_map_inline:nn
621 {

```

```

622 % Not type-specific options.
623 tpairsep ,
624 tlistsep ,
625 tlastsep ,
626 notesep ,
627 % Possibly type-specific options.
628 namefont ,
629 namesep ,
630 pairsep ,
631 listsep ,
632 lastsep ,
633 rangesep ,
634 reffont ,
635 refpre ,
636 refpos ,
637 reffont-in ,
638 refpre-in ,
639 refpos-in ,
640 }
641 {
642 \keys_define:nn { zref-clever }
643 {
644 #1 .default:V = \c_novalue_tl ,
645 #1 .code:n =
646 {
647 \tl_if_novalue:nTF {##1}
648 { \prop_remove:Nn \l__zrefclever_ref_options_prop {#1} }
649 { \prop_put:Nnn \l__zrefclever_ref_options_prop {#1} {##1} }
650 } ,
651 }
652 }

```

Package options

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

```

653 \RequirePackage { l3keys2e }
654 \ProcessKeysOptions { zref-clever }

```

`\zcsetup` Provide `\zcsetup`.

```

655 \NewDocumentCommand \zcsetup { m }
656 { \keys_set:nn { zref-clever } {#1} }

```

(End definition for \zcsetup.)

5 Type format

5.1 \zcRefTypeSetup

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

```

\l__zrefclever_setup_language_tl
657 \tl_new:N \l__zrefclever_setup_type_tl
658 \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`.

```

659 \NewDocumentCommand \zcRefTypeSetup { m m }
660 {
661   \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }
662   { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
663   \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
664   \keys_set:nn { zref-clever / typesetup } {#2}
665 }

```

(*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.

```

666 \clist_map_inline:nn
667 {
668   tpairsep ,
669   tlistsep ,
670   tlastsep ,
671   notesep ,
672 }
673 {
674   \keys_define:nn { zref-clever / typesetup }
675   {
676     #1 .code:n =
677     {
678       \msg_warning:nnn { zref-clever } { option-not-type-specific } {#1}
679     } ,
680   }
681 }

```

Possibly or necessarily type-specific options.

```

682 \clist_map_inline:nn
683 {
684   % Possibly type-specific options.
685   namefont ,
686   namesep ,
687   pairsep ,
688   listsep ,
689   lastsep ,
690   rangesep ,
691   reffont ,
692   refpre ,

```

```

693     refpos ,
694     reffont-in ,
695     refpre-in ,
696     refpos-in ,
697     % Necessarily type-specific options.
698     Name-sg ,
699     name-sg ,
700     Name-pl ,
701     name-pl ,
702     Name-sg-ab ,
703     name-sg-ab ,
704     Name-pl-ab ,
705     name-pl-ab ,
706 }
707 {
708   \keys_define:nn { zref-clever / typesetup }
709   {
710     #1 .default:V = \c_novalue_tl ,
711     #1 .code:n =
712     {
713       \tl_if_novalue:nTF {##1}
714       {
715         \prop_remove:cn
716         { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
717         {#1}
718       }
719       {
720         \prop_put:cnn
721         { l__zrefclever_type_ \l__zrefclever_setup_type_tl _options_prop }
722         {#1} {##1}
723       }
724     } ,
725   }
726 }

```

5.2 \zcDeclareTranslations

\zcDeclareTranslations Provide \zcDeclareTranslations.

```

727 \NewDocumentCommand \zcDeclareTranslations { m m }
728 {
729   \tl_set:Nn \l__zrefclever_setup_language_tl {#1}
730   \tl_clear:N \l__zrefclever_setup_type_tl
731   \keys_set:nn { zref-clever / translations } {#2}
732 }

```

(End definition for \zcDeclareTranslations.)

```

733 \keys_define:nn { zref-clever / translations }
734 {
735   type .code:n =
736   {
737     \tl_if_empty:nTF {#1}
738     { \tl_clear:N \l__zrefclever_setup_type_tl }
739     {
740       \prop_if_exist:cF { l__zrefclever_type_ #1 _options_prop }

```

```

741         { \prop_new:c { l__zrefclever_type_ #1 _options_prop } }
742         \tl_set:Nn \l__zrefclever_setup_type_tl {#1}
743     }
744 },
745 }

```

Not type-specific options.

```

746 \clist_map_inline:nn
747 {
748     tpairsep ,
749     tlistsep ,
750     tlastsep ,
751     notesep ,
752 }
753 {
754     \keys_define:nn { zref-clever / translations }
755     {
756         #1 .value_required:n = true ,
757         #1 .code:n =
758         {
759             \tl_if_empty:NTF \l__zrefclever_setup_type_tl
760             {
761                 \__zrefclever_declare_translation:xxn { \l__zrefclever_setup_language_tl }
762                 { zrefclever-default- #1 } {##1}
763             }
764             {
765                 \msg_warning:nnn { zref-clever }
766                 { option-not-type-specific } {#1}
767             }
768         } ,
769     }
770 }

```

Possibly type-specific options.

```

771 \clist_map_inline:nn
772 {
773     namesep ,
774     pairsep ,
775     listsep ,
776     lastsep ,
777     rangesep ,
778     refpre ,
779     refpos ,
780     refpre-in ,
781     refpos-in ,
782 }
783 {
784     \keys_define:nn { zref-clever / translations }
785     {
786         #1 .value_required:n = true ,
787         #1 .code:n =
788         {
789             \tl_if_empty:NTF \l__zrefclever_setup_type_tl
790             {
791                 \__zrefclever_declare_translation:xxn { \l__zrefclever_setup_language_tl }

```

```

792         { zrefclever-default- #1 } {##1}
793     }
794     {
795         \__zrefclever_declare_translation:xxn { \l__zrefclever_setup_language_tl }
796         { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
797     }
798 } ,
799 }
800 }

```

Necessarily type-specific options.

```

801 \clist_map_inline:nn
802 {
803     Name-sg ,
804     name-sg ,
805     Name-pl ,
806     name-pl ,
807     Name-sg-ab ,
808     name-sg-ab ,
809     Name-pl-ab ,
810     name-pl-ab ,
811 }
812 {
813     \keys_define:nn { zref-clever / translations }
814     {
815         #1 .value_required:n = true ,
816         #1 .code:n =
817         {
818             \tl_if_empty:NTF \l__zrefclever_setup_type_tl
819             {
820                 \msg_warning:nnn { zref-clever }
821                 { option-only-type-specific } {#1}
822             }
823             {
824                 \__zrefclever_declare_translation:xxn { \l__zrefclever_setup_language_tl }
825                 { zrefclever-type- \l__zrefclever_setup_type_tl - #1 } {##1}
826             }
827         } ,
828     }
829 }

```

6 \zcref

```

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

```

(End definition for \zcref.)

```

\l__zrefclever_zcref_labels_seq
\l__zrefclever_link_star_bool
832 \seq_new:N \l__zrefclever_zcref_labels_seq
833 \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}

834 \cs_new_protected:Npn \__zrefclever_zcref:nnn #1#2#3
835 {
836   \group_begin:
837     \keys_set:nn { zref-clever } {#3}
838     \seq_set_from_clist:Nn \l__zrefclever_zcref_labels_seq {#1}
839     \bool_set:Nn \l__zrefclever_link_star_bool {#2}
840     % Integration with 'zref-check'.
841     \bool_lazy_and:nnT
842       { \l__zrefclever_zrefcheck_available_bool }
843       { \l__zrefclever_zcref_with_check_bool }
844       { \zrefcheck_zcref_beg_label: }
845     \bool_lazy_or:nnT
846       { \l__zrefclever_typeset_sort_bool }
847       { \l__zrefclever_typeset_range_bool }
848       { \__zrefclever_sort_labels: }
849     \__zrefclever_typeset_refs:
850     % Typeset \texttt{note}.
851     \l__zrefclever_noteseq_tl
852     \l__zrefclever_zcref_note_tl
853     % Integration with 'zref-check'.
854     \bool_lazy_and:nnT
855       { \l__zrefclever_zrefcheck_available_bool }
856       { \l__zrefclever_zcref_with_check_bool }
857       {
858         \zrefcheck_zcref_end_label_maybe:
859         \zrefcheck_zcref_run_checks_on_labels:n
860         { \l__zrefclever_zcref_labels_seq }
861       }
862   \group_end:
863 }

```

(End definition for `__zrefclever_zcref:nnnn`.)

7 \zcpageref

```

\zcpageref \zcpageref{\(*)[\options]}{\labels}

864 \NewDocumentCommand \zcpageref { s O { } m }
865 {
866   \IfBooleanTF {#1}
867     { \zcref*[#2, ref = page] {#3} }
868     { \zcref [ #2, ref = page] {#3} }
869 }

```

(End definition for `\zcpageref`.)

8 Sorting

```
870 \int_new:N \l__zrefclever_sort_prior_a_int
871 \int_new:N \l__zrefclever_sort_prior_b_int
```

`\l__zrefclever_label_a_tl` `\l__zrefclever_label_b_tl` Aux variables, for use in sorting and typesetting. I could probably let go some of them in favor of `tmpa/tmpb`, but they do improve code readability.

```
\l__zrefclever_label_type_a_tl 872 \tl_new:N \l__zrefclever_label_a_tl
\l__zrefclever_label_type_b_tl 873 \tl_new:N \l__zrefclever_label_b_tl
\l__zrefclever_label_enclcnt_a_tl 874 \tl_new:N \l__zrefclever_label_type_a_tl
\l__zrefclever_label_enclcnt_b_tl 875 \tl_new:N \l__zrefclever_label_type_b_tl
\l__zrefclever_label_enclval_a_tl 876 \tl_new:N \l__zrefclever_label_enclcnt_a_tl
\l__zrefclever_label_enclval_b_tl 877 \tl_new:N \l__zrefclever_label_enclcnt_b_tl
878 \tl_new:N \l__zrefclever_label_enclval_a_tl
879 \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`.

```
880 \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.

```
881 \cs_new_protected:Npn \__zrefclever_sort_labels:
882 {
```

Store label types sequence.

```
883 \seq_clear:N \l__zrefclever_label_types_seq
884 \bool_if:NF \l__zrefclever_page_ref_bool
885 {
886 \seq_map_function:NN
887 \l__zrefclever_zcref_labels_seq \__zrefclever_label_type_put_new_right:n
888 }
```

Sort.

```
889 \seq_sort:Nn \l__zrefclever_zcref_labels_seq
890 {
891 \zref@ifrefundefined {##1}
892 {
893 \zref@ifrefundefined {##2}
894 {
895 % Neither label is defined.
896 \sort_return_same:
897 }
898 {
899 % The second label is defined, but the first isn't, leave the
900 % undefined first (to be more visible).
901 \sort_return_same:
```



```

902     }
903   }
904   {
905     \zref@ifrefundefined {##2}
906     {
907       % The first label is defined, but the second isn't, bring the
908       % second forward.
909       \sort_return_swapped:
910     }
911     {
912       % The interesting case: both labels are defined. The
913       % reference to the "default" property/counter or to the page
914       % are quite different from our perspective, they rely on
915       % different fields and even use different information for
916       % sorting, so we branch them here to specialized functions.
917       \bool_if:NTF \l__zrefclever_page_ref_bool
918         { \__zrefclever_sort_page:nn {##1} {##2} }
919         { \__zrefclever_sort_default:nn {##1} {##2} }
920     }
921   }
922 }
923 }

```

(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>}

924 \cs_new_protected:Npn \__zrefclever_label_type_put_new_right:n #1
925 {
926   \tl_set:Nx \l__zrefclever_label_type_a_tl
927   { \zref@extractdefault {#1} {zc@type} { \c_empty_tl } }
928   \tl_if_empty:NF \l__zrefclever_label_type_a_tl
929   {
930     \seq_if_in:NVF \l__zrefclever_label_types_seq \l__zrefclever_label_type_a_tl
931     {
932       \seq_put_right:NV
933       \l__zrefclever_label_types_seq \l__zrefclever_label_type_a_tl
934     }
935   }
936 }

```

(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.

```

937 \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.

```

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

```

(End definition for `\tl_reverse_items:V`. This function is documented on page ??.)

`__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}}

939 \cs_new_protected:Npn \__zrefclever_sort_default:nn #1#2
940 {
941   \tl_set:Nx \l__zrefclever_label_type_a_tl
942     { \zref@extractdefault {#1} {zc@type} { \c_empty_tl } }
943   \tl_set:Nx \l__zrefclever_label_type_b_tl
944     { \zref@extractdefault {#2} {zc@type} { \c_empty_tl } }
945
946   \bool_if:nTF
947     {
948     % The second label has a type, but the first doesn't, leave the
949     % undefined first (to be more visible).
950     \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
951     ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
952   }
953   { \sort_return_same: }
954   {
955     \bool_if:nTF
956       {
957       % The first label has a type, but the second doesn't, bring the
958       % second forward.
959       ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
960       \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
961     }
962     { \sort_return_swapped: }
963     {
964       \bool_if:nTF
965         {
966         % The interesting case: both labels have a type\dots{}
967         ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
968         ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
969       }
970       {
971       % Here we send this to a couple of auxiliary functions for no
972       % other reason than to keep this long function a little less
973       % unreadable.
974       \tl_if_eq:NNTF \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
975         {
976         % \dots{} and it's the same type.
977         \__zrefclever_sort_default_same_type:nn {#1} {#2}
978       }
979       {
980       % \dots{} and they are different types.
981       \__zrefclever_sort_default_different_types:nn {#1} {#2}
982     }

```

```

983     }
984     {
985         % Neither of the labels has a type. We can't do much of
986         % meaningful here, but if it's the same counter, compare it.
987         \exp_args:Nxx \tl_if_eq:nnTF
988         { \zref@extractdefault {#1} { counter } { } }
989         { \zref@extractdefault {#2} { counter } { } }
990         {
991             \int_compare:nNnTF
992             { \zref@extractdefault {#1} { zc@cntval } {-1} }
993             >
994             { \zref@extractdefault {#2} { zc@cntval } {-1} }
995             { \sort_return_swapped: }
996             { \sort_return_same: }
997         }
998         { \sort_return_same: }
999     }
1000 }
1001 }
1002 }

```

(End definition for `_zrefclever_sort_default:nn`.)

`_zrefclever_sort_default_same_type:nn`

```

1003 \cs_new_protected:Npn \_zrefclever_sort_default_same_type:nn #1#2
1004 {
1005     \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1006     { \zref@extractdefault {#1} { zc@enclcnt } { \c_empty_tl } }
1007     \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1008     { \tl_reverse_items:V \l__zrefclever_label_enclcnt_a_tl }
1009     \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1010     { \zref@extractdefault {#2} { zc@enclcnt } { \c_empty_tl } }
1011     \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1012     { \tl_reverse_items:V \l__zrefclever_label_enclcnt_b_tl }
1013     \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1014     { \zref@extractdefault {#1} { zc@enclval } { \c_empty_tl } }
1015     \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1016     { \tl_reverse_items:V \l__zrefclever_label_enclval_a_tl }
1017     \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1018     { \zref@extractdefault {#2} { zc@enclval } { \c_empty_tl } }
1019     \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1020     { \tl_reverse_items:V \l__zrefclever_label_enclval_b_tl }
1021
1022     \bool_set_false:N \l__zrefclever_sort_decided_bool
1023     % CHECK should I replace the tmp variables here?
1024     \tl_clear:N \l_tmpa_tl
1025     \tl_clear:N \l_tmpb_tl
1026     \bool_until_do:Nn \l__zrefclever_sort_decided_bool
1027     {
1028         \tl_set:Nx \l_tmpa_tl
1029         { \tl_head:N \l__zrefclever_label_enclcnt_a_tl }
1030         \tl_set:Nx \l_tmpb_tl
1031         { \tl_head:N \l__zrefclever_label_enclcnt_b_tl }
1032

```

```

1033 \bool_if:nTF
1034 {
1035   % Both are empty, meaning: neither labels have any (further)
1036   % ‘enclosing counters’ (left).
1037   \tl_if_empty_p:V \l_tmpa_tl &&
1038   \tl_if_empty_p:V \l_tmpb_tl
1039 }
1040 {
1041   \exp_args:Nxx \tl_if_eq:nTF
1042   { \zref@extractdefault {#1} { counter } { } }
1043   { \zref@extractdefault {#2} { counter } { } }
1044   {
1045     \bool_set_true:N \l__zrefclever_sort_decided_bool
1046     \int_compare:nNnTF
1047       { \zref@extractdefault {#1} { zc@cntval } {-1} }
1048       >
1049       { \zref@extractdefault {#2} { zc@cntval } {-1} }
1050       { \sort_return_swapped: }
1051       { \sort_return_same: }
1052   }
1053   {
1054     \msg_warning:nnnn { zref-clever }
1055     { counters-not-nested } {#1} {#2}
1056     \bool_set_true:N \l__zrefclever_sort_decided_bool
1057     \sort_return_same:
1058   }
1059 }
1060 {
1061   \bool_if:nTF
1062   {
1063     % ‘a’ is empty (and ‘b’ is not), meaning: ‘b’ is (possibly)
1064     % nested in ‘a’.
1065     \tl_if_empty_p:V \l_tmpa_tl
1066   }
1067   {
1068     \tl_set:Nx \l_tmpa_tl
1069     { {\zref@extractdefault {#1} { counter } { } } }
1070     \exp_args:NNx \tl_if_in:NnTF
1071     \l__zrefclever_label_enclcnt_b_tl { \l_tmpa_tl }
1072     {
1073       \bool_set_true:N \l__zrefclever_sort_decided_bool
1074       \sort_return_same:
1075     }
1076     {
1077       \msg_warning:nnnn { zref-clever }
1078       { counters-not-nested } {#1} {#2}
1079       \bool_set_true:N \l__zrefclever_sort_decided_bool
1080       \sort_return_same:
1081     }
1082   }
1083   {
1084     \bool_if:nTF
1085     {
1086       % ‘b’ is empty (and ‘a’ is not), meaning: ‘a’ is

```

```

1087 % (possibly) nested in 'b'.
1088 \tl_if_empty_p:V \l_tmpb_tl
1089 }
1090 {
1091   \tl_set:Nx \l_tmpb_tl
1092     { {\zref@extractdefault {#2} { counter } { }} }
1093   \exp_args:NNx \tl_if_in:NnTF
1094     \l__zrefclever_label_enclcnt_a_tl { \l_tmpb_tl }
1095     {
1096       \bool_set_true:N \l__zrefclever_sort_decided_bool
1097       \sort_return_swapped:
1098     }
1099     {
1100       \msg_warning:nnnn { zref-clever }
1101       { counters-not-nested } {#1} {#2}
1102       \bool_set_true:N \l__zrefclever_sort_decided_bool
1103       \sort_return_same:
1104     }
1105   }
1106   {
1107     % Neither is empty, meaning: we can (possibly) compare the
1108     % values of the current enclosing counter in the loop, if
1109     % they are equal, we are still in the loop, if they are
1110     % not, a sorting decision can be made directly.
1111     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
1112     {
1113       \int_compare:nNnTF
1114         { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1115         =
1116         { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1117         {
1118           \tl_set:Nx \l__zrefclever_label_enclcnt_a_tl
1119             { \tl_tail:N \l__zrefclever_label_enclcnt_a_tl }
1120           \tl_set:Nx \l__zrefclever_label_enclcnt_b_tl
1121             { \tl_tail:N \l__zrefclever_label_enclcnt_b_tl }
1122           \tl_set:Nx \l__zrefclever_label_enclval_a_tl
1123             { \tl_tail:N \l__zrefclever_label_enclval_a_tl }
1124           \tl_set:Nx \l__zrefclever_label_enclval_b_tl
1125             { \tl_tail:N \l__zrefclever_label_enclval_b_tl }
1126         }
1127         {
1128           \bool_set_true:N \l__zrefclever_sort_decided_bool
1129           \int_compare:nNnTF
1130             { \tl_head:N \l__zrefclever_label_enclval_a_tl }
1131             >
1132             { \tl_head:N \l__zrefclever_label_enclval_b_tl }
1133             { \sort_return_swapped: }
1134             { \sort_return_same: }
1135         }
1136     }
1137     {
1138       \msg_warning:nnnn { zref-clever }
1139       { counters-not-nested } {#1} {#2}
1140       \bool_set_true:N \l__zrefclever_sort_decided_bool

```

```

1141         \sort_return_same:
1142     }
1143 }
1144 }
1145 }
1146 }
1147 }

```

(End definition for _zrefclever_sort_default_same_type:nn.)

_zrefclever_sort_default_different_types:nn

```

1148 \cs_new_protected:Npn \_zrefclever_sort_default_different_types:nn #1#2
1149 {
1150     \int_zero:N \l_zrefclever_sort_prior_a_int
1151     \int_zero:N \l_zrefclever_sort_prior_b_int
1152     % \cs{l_zrefclever_typesort_seq} was stored in reverse sequence, and we compute
1153     % the sort priorities in the negative range, so that we can implicitly
1154     % rely on '0' being the 'last value'.
1155     \seq_map_indexed_inline:Nn \l_zrefclever_typesort_seq
1156     {
1157         \tl_if_eq:nnTF {##2} {{othertypes}}
1158         {
1159             \int_compare:nNnT { \l_zrefclever_sort_prior_a_int } = { 0 }
1160             { \int_set:Nn \l_zrefclever_sort_prior_a_int { - ##1 } }
1161             \int_compare:nNnT { \l_zrefclever_sort_prior_b_int } = { 0 }
1162             { \int_set:Nn \l_zrefclever_sort_prior_b_int { - ##1 } }
1163         }
1164         {
1165             \tl_if_eq:NnTF \l_zrefclever_label_type_a_tl {##2}
1166             { \int_set:Nn \l_zrefclever_sort_prior_a_int { - ##1 } }
1167             {
1168                 \tl_if_eq:NnTF \l_zrefclever_label_type_b_tl {##2}
1169                 { \int_set:Nn \l_zrefclever_sort_prior_b_int { - ##1 } }
1170             }
1171         }
1172     }
1173     \bool_if:nTF
1174     {
1175         \int_compare_p:nNn
1176         { \l_zrefclever_sort_prior_a_int } <
1177         { \l_zrefclever_sort_prior_b_int }
1178     }
1179     { \sort_return_same: }
1180     {
1181         \bool_if:nTF
1182         {
1183             \int_compare_p:nNn
1184             { \l_zrefclever_sort_prior_a_int } >
1185             { \l_zrefclever_sort_prior_b_int }
1186         }
1187         { \sort_return_swapped: }
1188         {
1189             % Sort priorities are equal for different types: the type that
1190             % occurs first in \meta{labels}, as given by the user, is kept (or

```

```

1191         % brought) forward.
1192         \seq_map_inline:Nn \l__zrefclever_label_types_seq
1193         {
1194             \tl_if_eq:NnTF \l__zrefclever_label_type_a_tl {##1}
1195             { \seq_map_break:n { \sort_return_same: } }
1196             {
1197                 \tl_if_eq:NnTF \l__zrefclever_label_type_b_tl {##1}
1198                 { \seq_map_break:n { \sort_return_swapped: } }
1199             }
1200         }
1201     }
1202 }
1203 }

```

(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}

1204 \cs_new_protected:Npn \__zrefclever_sort_page:nn #1#2
1205 {
1206     \int_compare:nNnTF
1207     { \zref@extractdefault {#1} { abspage } {-1} }
1208     >
1209     { \zref@extractdefault {#2} { abspage } {-1} }
1210     { \sort_return_swapped: }
1211     { \sort_return_same: }
1212 }

```

(End definition for `__zrefclever_sort_page:nn`.)

9 Typesetting

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

Typesetting variables

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

```
1213 \bool_new:N \l_zrefclever_typeset_last_bool
1214 \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.

```
1215 \seq_new:N \l_zrefclever_typeset_labels_seq
1216 \tl_new:N \l_zrefclever_typeset_queue_prev_tl
1217 \tl_new:N \l_zrefclever_typeset_queue_curr_tl
1218 \tl_new:N \l_zrefclever_type_first_label_tl
1219 \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.

```
1220 \int_new:N \l_zrefclever_label_count_int
1221 \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.

```
1222 \int_new:N \l_zrefclever_range_count_int
1223 \int_new:N \l_zrefclever_range_same_count_int
1224 \tl_new:N \l_zrefclever_range_beg_label_tl
1225 \bool_new:N \l_zrefclever_next_maybe_range_bool
1226 \bool_new:N \l_zrefclever_next_is_same_bool
1227 \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.

```

1228 \tl_new:N \l__zrefclever_namefont_tl
1229 \tl_new:N \l__zrefclever_reffont_out_tl
1230 \tl_new:N \l__zrefclever_reffont_in_tl
1231
1232 \tl_new:N \l__zrefclever_namesep_tl
1233 \tl_new:N \l__zrefclever_rangeseq_tl
1234 \tl_new:N \l__zrefclever_pairsep_tl
1235 \tl_new:N \l__zrefclever_listsep_tl
1236 \tl_new:N \l__zrefclever_lastsep_tl
1237 % ‘t’ for ‘type’
1238 \tl_new:N \l__zrefclever_tpairsep_tl
1239 \tl_new:N \l__zrefclever_tlistsep_tl
1240 \tl_new:N \l__zrefclever_tlastsep_tl
1241 \tl_new:N \l__zrefclever_noteseq_tl
1242 \tl_new:N \l__zrefclever_refpre_out_tl
1243 \tl_new:N \l__zrefclever_refpos_out_tl
1244 \tl_new:N \l__zrefclever_refpre_in_tl
1245 \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:`.

```

1246 \tl_new:N \l__zrefclever_type_name_tl
1247 \bool_new:N \l__zrefclever_name_in_link_bool
1248 \tl_new:N \l__zrefclever_name_format_tl
1249 \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`.

```

1250 \cs_new_protected:Npn \__zrefclever_typeset_refs:
1251 {
1252   \seq_set_eq:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_zceref_labels_seq
1253   \tl_clear:N \l__zrefclever_typeset_queue_prev_tl
1254   \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1255   \tl_clear:N \l__zrefclever_type_first_label_tl
1256   \tl_clear:N \l__zrefclever_type_first_label_type_tl
1257   \tl_clear:N \l__zrefclever_range_beg_label_tl
1258   \int_zero:N \l__zrefclever_label_count_int
1259   \int_zero:N \l__zrefclever_type_count_int
1260   \int_zero:N \l__zrefclever_range_count_int
1261   \int_zero:N \l__zrefclever_range_same_count_int
1262
1263   % Get not-type-specific separators and refpre/pos options.
1264   \__zrefclever_get_option_with_transl:nN {tpairsep} \l__zrefclever_tpairsep_tl
1265   \__zrefclever_get_option_with_transl:nN {tlistsep} \l__zrefclever_tlistsep_tl
1266   \__zrefclever_get_option_with_transl:nN {tlastsep} \l__zrefclever_tlastsep_tl
1267   \__zrefclever_get_option_with_transl:nN {noteseq} \l__zrefclever_noteseq_tl
1268

```

```

1269 % Set the font option for this zcref call.
1270 \l__zrefclever_ref_typeset_font_tl
1271
1272 % Loop over the label list in sequence.
1273 \bool_set_false:N \l__zrefclever_typeset_last_bool
1274 \bool_until_do:Nn \l__zrefclever_typeset_last_bool
1275 {
1276   \seq_pop_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_a_tl
1277   \seq_if_empty:NTF \l__zrefclever_typeset_labels_seq
1278   {
1279     \tl_clear:N \l__zrefclever_label_b_tl
1280     \bool_set_true:N \l__zrefclever_typeset_last_bool
1281   }
1282   { \seq_get_left:NN \l__zrefclever_typeset_labels_seq \l__zrefclever_label_b_tl }
1283
1284   \bool_if:NTF \l__zrefclever_page_ref_bool
1285   {
1286     \tl_set:Nn \l__zrefclever_label_type_a_tl { page }
1287     \tl_set:Nn \l__zrefclever_label_type_b_tl { page }
1288   }
1289   {
1290     \tl_set:Nx \l__zrefclever_label_type_a_tl
1291     {
1292       \zref@extractdefault
1293       { \l__zrefclever_label_a_tl } { zc@type } { \c_empty_tl }
1294     }
1295     \tl_set:Nx \l__zrefclever_label_type_b_tl
1296     {
1297       \zref@extractdefault
1298       { \l__zrefclever_label_b_tl } { zc@type } { \c_empty_tl }
1299     }
1300   }
1301
1302   % First, we establish whether the ‘current label’ (i.e. ‘a’) is the
1303   % last one of its type. This can happen because the ‘next label’
1304   % (i.e. ‘b’) is of a different type (or different definition status),
1305   % or because we are at the end of the list.
1306   \bool_if:NTF \l__zrefclever_typeset_last_bool
1307   { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1308   {
1309     \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1310     {
1311       \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1312       { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1313       { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1314     }
1315     {
1316       \zref@ifrefundefined { \l__zrefclever_label_b_tl }
1317       { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1318       {
1319         % Neither is undefined, we must check the types.
1320         \bool_if:nTF
1321         % Both empty: same ‘type’.
1322         {

```

```

1323         \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1324         \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1325     }
1326     { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1327     {
1328         \bool_if:nTF
1329         % Neither empty: compare types.
1330         {
1331             ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl &&
1332             ! \tl_if_empty_p:N \l__zrefclever_label_type_b_tl
1333         }
1334         {
1335             \tl_if_eq:NNTF
1336             \l__zrefclever_label_type_a_tl \l__zrefclever_label_type_b_tl
1337             { \bool_set_false:N \l__zrefclever_last_of_type_bool }
1338             { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1339         }
1340         % One empty, the other not: different ‘types’.
1341         { \bool_set_true:N \l__zrefclever_last_of_type_bool }
1342     }
1343 }
1344 }
1345 }
1346
1347 % Handle warnings in case of reference or type undefined.
1348 \zref@refused { \l__zrefclever_label_a_tl }
1349 \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1350 {}
1351 {
1352     \tl_if_empty:NT \l__zrefclever_label_type_a_tl
1353     {
1354         \msg_warning:nxx { zref-clever } { missing-type }
1355         { \l__zrefclever_label_a_tl }
1356     }
1357 }
1358
1359 % Get type-specific separators, refpre/pos and font options, once per
1360 % type.
1361 \int_compare:nNnT { \l__zrefclever_label_count_int } = { 0 }
1362 {
1363     \__zrefclever_get_option_plain:nN {namefont}         \l__zrefclever_namefont_tl
1364     \__zrefclever_get_option_plain:nN {reffont}         \l__zrefclever_reffont_out_tl
1365     \__zrefclever_get_option_plain:nN {reffont-in}      \l__zrefclever_reffont_in_tl
1366     \__zrefclever_get_option_with_transl:nN {namesep}   \l__zrefclever_namesep_tl
1367     \__zrefclever_get_option_with_transl:nN {rangesep} \l__zrefclever_rangesep_tl
1368     \__zrefclever_get_option_with_transl:nN {pairsep}   \l__zrefclever_pairsep_tl
1369     \__zrefclever_get_option_with_transl:nN {listsep}   \l__zrefclever_listsep_tl
1370     \__zrefclever_get_option_with_transl:nN {lastsep}   \l__zrefclever_lastsep_tl
1371     \__zrefclever_get_option_with_transl:nN {refpre}     \l__zrefclever_refpre_out_tl
1372     \__zrefclever_get_option_with_transl:nN {refpos}     \l__zrefclever_refpos_out_tl
1373     \__zrefclever_get_option_with_transl:nN {refpre-in} \l__zrefclever_refpre_in_tl
1374     \__zrefclever_get_option_with_transl:nN {refpos-in} \l__zrefclever_refpos_in_tl
1375 }
1376

```

```

1377      % Here we send this to a couple of auxiliary functions for no other
1378      % reason than to keep this long function a little less unreadable.
1379      \bool_if:NTF \l__zrefclever_last_of_type_bool
1380      {
1381          % There exists no next label of the same type as the current.
1382          \__zrefclever_typeset_refs_aux_last_of_type:
1383      }
1384      {
1385          % There exists a next label of the same type as the current.
1386          \__zrefclever_typeset_refs_aux_not_last_of_type:
1387      }
1388  }
1389 }

```

(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.

```

1390 \cs_new_protected:Npn \__zrefclever_typeset_refs_aux_last_of_type:
1391 {
1392     % Process the current label to the current queue.
1393     \int_case:nnF { \l__zrefclever_label_count_int }
1394     {
1395         % It is the last label of its type, but also the first one, and that's
1396         % what matters here: just store it.
1397         { 0 }
1398         {
1399             \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1400             \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1401         }
1402
1403         % The last is the second: we have a pair (if not repeated).
1404         { 1 }
1405         {
1406             \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1407             {
1408                 \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1409                 {
1410                     \exp_not:V \l__zrefclever_pairsep_tl
1411                     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1412                 }
1413             }
1414         }
1415     }
1416     % If neither the first, nor the second: we have the last label
1417     % on the current type list (if not repeated).
1418     {
1419         \int_case:nnF { \l__zrefclever_range_count_int }
1420         {
1421             % There was no range going on.
1422             {0}
1423             {
1424                 \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1425                 {
1426                     \exp_not:V \l__zrefclever_lastsep_tl

```

```

1427         \_zrefclever_get_ref:V \l__zrefclever_label_a_tl
1428     }
1429 }
1430 % Last in the range is also the second in it.
1431 {1}
1432 {
1433     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1434     {
1435         % We know 'range_beg_label' is not empty, since this is the
1436         % second element in the range, but the third or more in the
1437         % type list.
1438         \exp_not:V \l__zrefclever_listsep_tl
1439         \_zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1440         \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1441         {
1442             \exp_not:V \l__zrefclever_lastsep_tl
1443             \_zrefclever_get_ref:V \l__zrefclever_label_a_tl
1444         }
1445     }
1446 }
1447 }
1448 % Last in the range is third or more in it.
1449 {
1450     \int_case:nnF
1451     { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1452     {
1453         % Repetition, not a range.
1454         {0}
1455         {
1456             % If 'range_beg_label' is empty, it means it was also the
1457             % first of the type, and hence was already handled.
1458             \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1459             {
1460                 \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1461                 {
1462                     \exp_not:V \l__zrefclever_lastsep_tl
1463                     \_zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1464                 }
1465             }
1466         }
1467         % A "range", but with no skipped value, treat as list.
1468         {1}
1469         {
1470             \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1471             {
1472                 % Ditto.
1473                 \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1474                 {
1475                     \exp_not:V \l__zrefclever_listsep_tl
1476                     \_zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1477                 }
1478                 \exp_not:V \l__zrefclever_lastsep_tl
1479                 \_zrefclever_get_ref:V \l__zrefclever_label_a_tl
1480             }

```

```

1481     }
1482   }
1483   {
1484     % An actual range.
1485     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1486     {
1487       % Ditto.
1488       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1489       {
1490         \exp_not:V \l__zrefclever_lastsep_tl
1491         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1492       }
1493       \exp_not:V \l__zrefclever_rangesep_tl
1494       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1495     }
1496   }
1497 }
1498 }
1499
1500 % Handle ‘‘range’’ option. The idea is simple: if the queue is not empty,
1501 % we replace it with the end of the range (or pair). We can still
1502 % retrieve the end of the range from \cs{l__zrefclever_label_a_tl} since we know to
1503 % be processing the last label of its type at this point.
1504 \bool_if:NT \l__zrefclever_typeset_range_bool
1505 {
1506   \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1507   {
1508     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1509     { }
1510     {
1511       \msg_warning:nxx { zref-clever } { single-element-range }
1512       { \l__zrefclever_type_first_label_type_tl }
1513     }
1514   }
1515   {
1516     \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1517     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1518     { }
1519     {
1520       \__zrefclever_labels_in_sequence:nn
1521       { \l__zrefclever_type_first_label_tl } { \l__zrefclever_label_a_tl }
1522     }
1523     \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1524     {
1525       \bool_if:NTF \l__zrefclever_next_maybe_range_bool
1526       { \exp_not:V \l__zrefclever_pairsep_tl }
1527       { \exp_not:V \l__zrefclever_rangesep_tl }
1528       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1529     }
1530   }
1531 }
1532
1533 % Now that the type is finished, we can add the name and the first ref to
1534 % the queue. Or, if ‘‘typset’’ option is not ‘‘both’’, handle it here

```

```

1535 % too.
1536 \__zrefclever_type_name_setup:
1537 \bool_if:nTF
1538 { \l__zrefclever_typeset_ref_bool && \l__zrefclever_typeset_name_bool }
1539 {
1540   \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1541   { \__zrefclever_get_ref_first: }
1542 }
1543 {
1544   \bool_if:nTF
1545   { \l__zrefclever_typeset_ref_bool }
1546   {
1547     \tl_put_left:Nx \l__zrefclever_typeset_queue_curr_tl
1548     { \__zrefclever_get_ref:V \l__zrefclever_type_first_label_tl }
1549   }
1550   {
1551     \bool_if:nTF
1552     { \l__zrefclever_typeset_name_bool }
1553     {
1554       \tl_set:Nx \l__zrefclever_typeset_queue_curr_tl
1555       {
1556         \bool_if:NTF \l__zrefclever_name_in_link_bool
1557         {
1558           \exp_not:N \group_begin:
1559           \exp_not:V \l__zrefclever_namefont_tl
1560           % It's two '@s', but escaped for DocStrip.
1561           \exp_not:N \hyper@@link
1562           {
1563             \zref@ifrefcontainsprop
1564             { \l__zrefclever_type_first_label_tl } { urluse }
1565             {
1566               \zref@extractdefault
1567               { \l__zrefclever_type_first_label_tl }
1568               { urluse } {}
1569             }
1570             {
1571               \zref@extractdefault
1572               { \l__zrefclever_type_first_label_tl }
1573               { url } {}
1574             }
1575           }
1576           {
1577             \zref@extractdefault
1578             { \l__zrefclever_type_first_label_tl } { anchor } {}
1579           }
1580           { \exp_not:V \l__zrefclever_type_name_tl }
1581           \exp_not:N \group_end:
1582         }
1583         {
1584           \exp_not:N \group_begin:
1585           \exp_not:V \l__zrefclever_namefont_tl
1586           \exp_not:V \l__zrefclever_type_name_tl
1587           \exp_not:N \group_end:
1588         }

```

```

1589         }
1590     }
1591     {
1592         % This case would correspond to "typeset=none" but should not
1593         % happen, given the options are set up to typeset at least one
1594         % of "ref" or "name", but a sensible fallback, equal to the
1595         % behavior of ‘‘both’’.
1596         \tl_put_left:Nx
1597             \l__zrefclever_typeset_queue_curr_tl { \__zrefclever_get_ref_first: }
1598     }
1599 }
1600 }
1601
1602 % Typeset the previous type, if there is one.
1603 \int_compare:nNnT { \l__zrefclever_type_count_int } > { 0 }
1604 {
1605     \int_compare:nNnT { \l__zrefclever_type_count_int } > { 1 }
1606     { \l__zrefclever_tlistsep_tl }
1607     \l__zrefclever_typeset_queue_prev_tl
1608 }
1609
1610 % Wrap up loop, or prepare for next iteration.
1611 \bool_if:NTF \l__zrefclever_typeset_last_bool
1612 {
1613     % We are finishing, typeset the current queue.
1614     \int_case:nnF { \l__zrefclever_type_count_int }
1615     {
1616         % Single type.
1617         { 0 }
1618         { \l__zrefclever_typeset_queue_curr_tl }
1619         % Pair of types.
1620         { 1 }
1621         {
1622             \l__zrefclever_tpairsep_tl
1623             \l__zrefclever_typeset_queue_curr_tl
1624         }
1625     }
1626     {
1627         % Last in list of types.
1628         \l__zrefclever_tlastsep_tl
1629         \l__zrefclever_typeset_queue_curr_tl
1630     }
1631 }
1632 {
1633     % There are further labels, set variables for next iteration.
1634     \tl_set_eq:NN
1635         \l__zrefclever_typeset_queue_prev_tl \l__zrefclever_typeset_queue_curr_tl
1636     \tl_clear:N \l__zrefclever_typeset_queue_curr_tl
1637     \tl_clear:N \l__zrefclever_type_first_label_tl
1638     \tl_clear:N \l__zrefclever_type_first_label_type_tl
1639     \tl_clear:N \l__zrefclever_range_beg_label_tl
1640     \int_zero:N \l__zrefclever_label_count_int
1641     \int_incr:N \l__zrefclever_type_count_int
1642     \int_zero:N \l__zrefclever_range_count_int

```



```

1643         \int_zero:N \l__zrefclever_range_same_count_int
1644     }
1645 }

```

(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.

```

1646 \cs_new_protected:Npn \__zrefclever_typeset_refs_aux_not_last_of_type:
1647 {
1648     % Signal if next label may form a range with the current one (of
1649     % course, only considered if compression is enabled in the first
1650     % place).
1651     \bool_set_false:N \l__zrefclever_next_maybe_range_bool
1652     \bool_set_false:N \l__zrefclever_next_is_same_bool
1653     \bool_lazy_and:nnT
1654     { \l__zrefclever_typeset_compress_bool }
1655     % Currently no-op, but kept as ‘‘handle’’ to inhibit compression of
1656     % individual labels.
1657     { ! \l__zrefclever_range_inhibit_next_bool }
1658     {
1659         \zref@ifrefundefined { \l__zrefclever_label_a_tl }
1660         { }
1661         {
1662             \__zrefclever_labels_in_sequence:nn
1663             { \l__zrefclever_label_a_tl } { \l__zrefclever_label_b_tl }
1664         }
1665     }
1666
1667     % Process the current label to the current queue.
1668     \int_compare:nNnTF { \l__zrefclever_label_count_int } = { 0 }
1669     {
1670         % Current label is the first of its type (also not the last, but it
1671         % doesn’t matter here): just store the label.
1672         \tl_set:NV \l__zrefclever_type_first_label_tl \l__zrefclever_label_a_tl
1673         \tl_set:NV \l__zrefclever_type_first_label_type_tl \l__zrefclever_label_type_a_tl
1674
1675         % If the next label may be part of a range, we set ‘range_beg_label’
1676         % to ‘‘empty’’ (we deal with it as the ‘‘first’’, and must do it
1677         % there, to handle hyperlinking), but also step the range counters.
1678         \bool_if:NT \l__zrefclever_next_maybe_range_bool
1679         {
1680             \tl_clear:N \l__zrefclever_range_beg_label_tl
1681             \int_incr:N \l__zrefclever_range_count_int
1682             \bool_if:NT \l__zrefclever_next_is_same_bool
1683             { \int_incr:N \l__zrefclever_range_same_count_int }
1684         }
1685     }
1686     {
1687         % Current label is neither the first (nor the last) of its
1688         % type.
1689         \bool_if:NTF \l__zrefclever_next_maybe_range_bool
1690         {
1691             % Starting, or continuing a range.
1692             \int_compare:nNnTF

```

```

1693 { \l__zrefclever_range_count_int } = {0}
1694 {
1695   % There was no range going, we are starting one.
1696   \tl_set:NV \l__zrefclever_range_beg_label_tl \l__zrefclever_label_a_tl
1697   \int_incr:N \l__zrefclever_range_count_int
1698   \bool_if:NT \l__zrefclever_next_is_same_bool
1699     { \int_incr:N \l__zrefclever_range_same_count_int }
1700 }
1701 {
1702   % Second or more in the range, but not the last.
1703   \int_incr:N \l__zrefclever_range_count_int
1704   \bool_if:NT \l__zrefclever_next_is_same_bool
1705     { \int_incr:N \l__zrefclever_range_same_count_int }
1706 }
1707 }
1708 {
1709   % Next element is not in sequence, meaning: there was no range, or
1710   % we are closing one.
1711   \int_case:nnF { \l__zrefclever_range_count_int }
1712   {
1713     % There was no range going on.
1714     {0}
1715     {
1716       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1717       {
1718         \exp_not:V \l__zrefclever_listsep_tl
1719         \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1720       }
1721     }
1722     % Last is second in the range: if 'range_same_count' is also
1723     % '1', it's a repetition (drop it), otherwise, it's a "pair
1724     % within a list", treat as list.
1725     {1}
1726     {
1727       \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1728       {
1729         \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1730         {
1731           \exp_not:V \l__zrefclever_listsep_tl
1732           \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1733         }
1734         \int_compare:nNnF { \l__zrefclever_range_same_count_int } = {1}
1735         {
1736           \exp_not:V \l__zrefclever_listsep_tl
1737           \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1738         }
1739       }
1740     }
1741   }
1742   {
1743     % Last is third or more in the range: if 'range_count' and
1744     % 'range_same_count' are the same, its a repetition (drop it),
1745     % if they differ by '1', its a list, if they differ by more,
1746     % it is a real range.

```

```

1747 \int_case:nnF
1748 { \l__zrefclever_range_count_int - \l__zrefclever_range_same_count_int }
1749 {
1750   {0}
1751   {
1752     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1753     {
1754       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1755       {
1756         \exp_not:V \l__zrefclever_listsep_tl
1757         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1758       }
1759     }
1760   }
1761   {1}
1762   {
1763     \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1764     {
1765       \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1766       {
1767         \exp_not:V \l__zrefclever_listsep_tl
1768         \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1769       }
1770       \exp_not:V \l__zrefclever_listsep_tl
1771       \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1772     }
1773   }
1774 }
1775 {
1776   \tl_put_right:Nx \l__zrefclever_typeset_queue_curr_tl
1777   {
1778     \tl_if_empty:VF \l__zrefclever_range_beg_label_tl
1779     {
1780       \exp_not:V \l__zrefclever_listsep_tl
1781       \__zrefclever_get_ref:V \l__zrefclever_range_beg_label_tl
1782     }
1783     \exp_not:V \l__zrefclever_rangesep_tl
1784     \__zrefclever_get_ref:V \l__zrefclever_label_a_tl
1785   }
1786 }
1787 }
1788 % Reset counters.
1789 \int_zero:N \l__zrefclever_range_count_int
1790 \int_zero:N \l__zrefclever_range_same_count_int
1791 }
1792 }
1793 % Step label counter for next iteration.
1794 \int_incr:N \l__zrefclever_label_count_int
1795 }

```

(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.

```

1796 \cs_new:Npn \_zrefclever_get_ref:n #1
1797 {
1798   \zref@ifrefcontainsprop {#1} { \l__zrefclever_ref_property_tl }
1799   {
1800     \bool_if:nTF
1801       { \l__zrefclever_use_hyperref_bool && ! \l__zrefclever_link_star_bool }
1802       {
1803         \exp_not:N \group_begin:
1804         \exp_not:V \l__zrefclever_reffont_out_tl
1805         \exp_not:V \l__zrefclever_refpre_out_tl
1806         \exp_not:N \group_begin:
1807         \exp_not:V \l__zrefclever_reffont_in_tl
1808         % It's two '@s', but escaped for DocStrip.
1809         \exp_not:N \hyper@@link
1810         {
1811           \zref@ifrefcontainsprop {#1} { urluse }
1812           { \zref@extractdefault {#1} { urluse } {} }
1813           { \zref@extractdefault {#1} { url } {} }
1814         }
1815         { \zref@extractdefault {#1} { anchor } {} }
1816         {
1817           \exp_not:V \l__zrefclever_refpre_in_tl
1818           \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1819           \exp_not:V \l__zrefclever_refpos_in_tl
1820         }
1821         \exp_not:N \group_end:
1822         \exp_not:V \l__zrefclever_refpos_out_tl
1823         \exp_not:N \group_end:
1824       }
1825       {
1826         \exp_not:N \group_begin:
1827         \exp_not:V \l__zrefclever_reffont_out_tl
1828         \exp_not:V \l__zrefclever_refpre_out_tl
1829         \exp_not:N \group_begin:
1830         \exp_not:V \l__zrefclever_reffont_in_tl
1831         \exp_not:V \l__zrefclever_refpre_in_tl
1832         \zref@extractdefault {#1} { \l__zrefclever_ref_property_tl } {}
1833         \exp_not:V \l__zrefclever_refpos_in_tl
1834         \exp_not:N \group_end:
1835         \exp_not:V \l__zrefclever_refpos_out_tl
1836         \exp_not:N \group_end:
1837       }
1838     }
1839     { \exp_not:N \zref@default }
1840   }
1841 \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.

```

1842 \cs_new_protected:Npn \_zrefclever_type_name_setup:
1843 {
1844   \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1845   { \tl_clear:N \l__zrefclever_type_name_tl }
1846   {
1847     \tl_if_empty:NTF \l__zrefclever_type_first_label_type_tl
1848     { \tl_clear:N \l__zrefclever_type_name_tl }
1849     {

```

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

```

1850   \bool_lazy_or:nnTF
1851   { \l__zrefclever_capitalize_bool }
1852   {
1853     \l__zrefclever_capitalize_first_bool &&
1854     \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1855   }
1856   { \tl_set:Nn \l__zrefclever_name_format_tl {Name} }
1857   { \tl_set:Nn \l__zrefclever_name_format_tl {name} }
1858   % If the queue is empty, we have a singular, otherwise, plural.
1859   \tl_if_empty:NTF \l__zrefclever_typeset_queue_curr_tl
1860   { \tl_put_right:Nn \l__zrefclever_name_format_tl { -sg } }
1861   { \tl_put_right:Nn \l__zrefclever_name_format_tl { -pl } }
1862   \bool_lazy_and:nnTF
1863   { \l__zrefclever_abbrev_bool }
1864   {
1865     ! \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 } ||
1866     ! \l__zrefclever_noabbrev_first_bool
1867   }
1868   {
1869     \tl_set:NV \l__zrefclever_name_format_fallback_tl \l__zrefclever_name_format
1870     \tl_put_right:Nn \l__zrefclever_name_format_tl { -ab }
1871   }
1872   { \tl_clear:N \l__zrefclever_name_format_fallback_tl }
1873
1874   \tl_if_empty:NTF \l__zrefclever_name_format_fallback_tl
1875   {
1876     \prop_get:cVNF
1877     { \l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1878       \l__zrefclever_name_format_tl
1879       \l__zrefclever_type_name_tl
1880     }
1881     \_zrefclever_if_translation:xxTF
1882     { \l__zrefclever_ref_language_tl }
1883     {
1884       zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1885       \l__zrefclever_name_format_tl
1886     }
1887     {
1888       \_zrefclever_get_translation_for:nxx { \l__zrefclever_type_name_tl
1889       { \l__zrefclever_ref_language_tl }

```

```

1890         {
1891             zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1892             \l__zrefclever_name_format_tl
1893         }
1894     }
1895     {
1896         \tl_clear:N \l__zrefclever_type_name_tl
1897         \msg_warning:nnx { zref-clever } { missing-name }
1898         { \l__zrefclever_type_first_label_type_tl }
1899     }
1900 }
1901 }
1902 {
1903     \prop_get:cVNF
1904     { \l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options_pro
1905       \l__zrefclever_name_format_tl
1906       \l__zrefclever_type_name_tl
1907     {
1908         \prop_get:cVNF
1909         { \l__zrefclever_type_ \l__zrefclever_type_first_label_type_tl _options
1910           \l__zrefclever_name_format_fallback_tl
1911           \l__zrefclever_type_name_tl
1912         {
1913             \__zrefclever_if_translation:xxTF
1914             { \l__zrefclever_ref_language_tl }
1915             {
1916                 zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1917                 \l__zrefclever_name_format_tl
1918             }
1919             {
1920                 \__zrefclever_get_translation_for:nxx { \l__zrefclever_type_name
1921                   { \l__zrefclever_ref_language_tl }
1922                   {
1923                       zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1924                       \l__zrefclever_name_format_tl
1925                   }
1926             }
1927             {
1928                 \__zrefclever_if_translation:xxTF
1929                 { \l__zrefclever_ref_language_tl }
1930                 {
1931                     zrefclever-type- \l__zrefclever_type_first_label_type_tl -
1932                     \l__zrefclever_name_format_fallback_tl
1933                 }
1934                 {
1935                     \__zrefclever_get_translation_for:nxx { \l__zrefclever_type_
1936                       { \l__zrefclever_ref_language_tl }
1937                       {
1938                           zrefclever-type- \l__zrefclever_type_first_label_type_tl
1939                           \l__zrefclever_name_format_fallback_tl
1940                       }
1941                 }
1942             }
1943             {
1944                 \tl_clear:N \l__zrefclever_type_name_tl

```

```

1944         \msg_warning:nxx { zref-clever } { missing-name }
1945         { \l__zrefclever_type_first_label_type_tl }
1946     }
1947 }
1948 }
1949 }
1950 }
1951 }
1952 }

```

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

```

1953 \bool_lazy_any:nTF
1954 {
1955     { ! \l__zrefclever_use_hyperref_bool }
1956     { \l__zrefclever_link_star_bool }
1957     { \tl_if_empty_p:N \l__zrefclever_type_name_tl }
1958     { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { false } }
1959 }
1960 { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1961 {
1962     \bool_lazy_any:nTF
1963     {
1964         { \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { true } }
1965         {
1966             \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { tsingle } &&
1967             \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl
1968         }
1969         {
1970             \str_if_eq_p:Vn \l__zrefclever_nameinlink_str { single } &&
1971             \tl_if_empty_p:N \l__zrefclever_typeset_queue_curr_tl &&
1972             \l__zrefclever_typeset_last_bool &&
1973             \int_compare_p:nNn { \l__zrefclever_type_count_int } = { 0 }
1974         }
1975     }
1976     { \bool_set_true:N \l__zrefclever_name_in_link_bool }
1977     { \bool_set_false:N \l__zrefclever_name_in_link_bool }
1978 }
1979 }

```

(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.

```

1980 \cs_new:Npn \__zrefclever_get_ref_first:
1981 {
1982     \zref@ifrefundefined { \l__zrefclever_type_first_label_tl }
1983     { \exp_not:N \zref@default }
1984     {
1985         \bool_if:NTF \l__zrefclever_name_in_link_bool
1986         {
1987             \zref@ifrefcontainsprop
1988             { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
1989             {
1990                 % It's two '@s', but escaped for DocStrip.

```

```

1991 \exp_not:N \hyper@@link
1992 {
1993   \zref@ifrefcontainsprop
1994   { \l__zrefclever_type_first_label_tl } { urluse }
1995   {
1996     \zref@extractdefault { \l__zrefclever_type_first_label_tl }
1997     { urluse } {}
1998   }
1999   {
2000     \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2001     { url } {}
2002   }
2003 }
2004 {
2005   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2006   { anchor } {}
2007 }
2008 {
2009   \exp_not:N \group_begin:
2010   \exp_not:V \l__zrefclever_namefont_tl
2011   \exp_not:V \l__zrefclever_type_name_tl
2012   \exp_not:N \group_end:
2013   \exp_not:V \l__zrefclever_namesep_tl
2014   \exp_not:N \group_begin:
2015   \exp_not:V \l__zrefclever_reffont_out_tl
2016   \exp_not:V \l__zrefclever_refpre_out_tl
2017   \exp_not:N \group_begin:
2018   \exp_not:V \l__zrefclever_reffont_in_tl
2019   \exp_not:V \l__zrefclever_refpre_in_tl
2020   \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2021   { \l__zrefclever_ref_property_tl } {}
2022   \exp_not:V \l__zrefclever_refpos_in_tl
2023   \exp_not:N \group_end:
2024   % hyperlink makes it's own group, we'd like to close the
2025   % 'refpre-out' group after 'refpos-out', but... we close
2026   % it here, and give the trailing 'refpos-out' its own
2027   % group. This will result that formatting given to
2028   % 'refpre-out' will not reach 'refpos-out', but I see no
2029   % alternative, and this has to be handled specially.
2030   \exp_not:N \group_end:
2031 }
2032 \exp_not:N \group_begin:
2033 % Ditto: special treatment.
2034 \exp_not:V \l__zrefclever_reffont_out_tl
2035 \exp_not:V \l__zrefclever_refpos_out_tl
2036 \exp_not:N \group_end:
2037 }
2038 {
2039   \exp_not:N \group_begin:
2040   \exp_not:V \l__zrefclever_namefont_tl
2041   \exp_not:V \l__zrefclever_type_name_tl
2042   \exp_not:N \group_end:
2043   \exp_not:V \l__zrefclever_namesep_tl
2044   \exp_not:N \zref@default

```



```

2045     }
2046   }
2047   {
2048     \tl_if_empty:NTF \l__zrefclever_type_name_tl
2049     {
2050       \exp_not:N \zref@default
2051       \exp_not:V \l__zrefclever_namesep_tl
2052     }
2053     {
2054       \exp_not:N \group_begin:
2055       \exp_not:V \l__zrefclever_namefont_tl
2056       \exp_not:V \l__zrefclever_type_name_tl
2057       \exp_not:N \group_end:
2058       \exp_not:V \l__zrefclever_namesep_tl
2059     }
2060   \zref@ifrefcontainsprop
2061   { \l__zrefclever_type_first_label_tl } { \l__zrefclever_ref_property_tl }
2062   {
2063     \bool_if:nTF
2064     {
2065       \l__zrefclever_use_hyperref_bool &&
2066       ! \l__zrefclever_link_star_bool
2067     }
2068     {
2069       \exp_not:N \group_begin:
2070       \exp_not:V \l__zrefclever_reffont_out_tl
2071       \exp_not:V \l__zrefclever_refpre_out_tl
2072       \exp_not:N \group_begin:
2073       \exp_not:V \l__zrefclever_reffont_in_tl
2074       % It's two '@s', but escaped for DocStrip.
2075       \exp_not:N \hyper@@link
2076       {
2077         \zref@ifrefcontainsprop
2078         { \l__zrefclever_type_first_label_tl } { urluse }
2079         {
2080           \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2081           { urluse } {}
2082         }
2083         {
2084           \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2085           { url } {}
2086         }
2087       }
2088     }
2089     {
2090       \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2091       { anchor } {}
2092     }
2093     {
2094       \exp_not:V \l__zrefclever_refpre_in_tl
2095       \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2096       { \l__zrefclever_ref_property_tl } {}
2097       \exp_not:V \l__zrefclever_refpos_in_tl
2098     }
2099   \exp_not:N \group_end:

```

```

2099         \exp_not:V \l__zrefclever_refpos_out_tl
2100         \exp_not:N \group_end:
2101     }
2102     {
2103         \exp_not:N \group_begin:
2104         \exp_not:V \l__zrefclever_reffont_out_tl
2105         \exp_not:V \l__zrefclever_refpre_out_tl
2106         \exp_not:N \group_begin:
2107         \exp_not:V \l__zrefclever_reffont_in_tl
2108         \exp_not:V \l__zrefclever_refpre_in_tl
2109         \zref@extractdefault { \l__zrefclever_type_first_label_tl }
2110             { \l__zrefclever_ref_property_tl } {}
2111         \exp_not:V \l__zrefclever_refpos_in_tl
2112         \exp_not:N \group_end:
2113         \exp_not:V \l__zrefclever_refpos_out_tl
2114         \exp_not:N \group_end:
2115     }
2116 }
2117 { \exp_not:N \zref@default }
2118 }
2119 }
2120 }

```

(End definition for __zrefclever_get_ref_first:.)

_zrefclever_get_option_with_transl:nN

```

2121 % \Arg{option} \Arg{var to store result}
2122 \cs_new_protected:Npn \__zrefclever_get_option_with_transl:nN #1#2
2123 {
2124     % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.
2125     \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2126     {
2127         % If not found, try the type specific options.
2128         \bool_lazy_all:nTF
2129         {
2130             { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2131             {
2132                 \prop_if_exist_p:c
2133                 { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2134             }
2135             {
2136                 \prop_if_in_p:cn
2137                 { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1}
2138             }
2139         }
2140         {
2141             \prop_get:cnN
2142             { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2143         }
2144         {
2145             % If not found, try the type specific translations.
2146             \__zrefclever_if_translation:xxTF
2147             { \l__zrefclever_ref_language_tl }
2148             { zrefclever-type- \l__zrefclever_label_type_a_tl - #1 }

```

```

2149     {
2150       \__zrefclever_get_translation_for:nxx {#2}
2151       { \l__zrefclever_ref_language_tl }
2152       { zrefclever-type- \l__zrefclever_label_type_a_tl - #1 }
2153     }
2154     {
2155       % If not found, try general translations. We are not
2156       % controlling for their existence, but we must make sure all
2157       % options being retrieved with
2158       % \cs{__zrefclever_get_option_with_transl:nN} have their values set for
2159       % 'English' and 'fallback'.
2160       \__zrefclever_get_translation_for:nxx {#2}
2161       { \l__zrefclever_ref_language_tl }
2162       { zrefclever-default- #1 }
2163     }
2164   }
2165 }
2166 }

```

(End definition for __zrefclever_get_option_with_transl:nN.)

__zrefclever_get_option_plain:nN

```

2167 \cs_new_protected:Npn \__zrefclever_get_option_plain:nN #1#2
2168 {
2169   % First attempt options stored in \cs{l__zrefclever_ref_options_prop}.
2170   \prop_get:NnNF \l__zrefclever_ref_options_prop {#1} #2
2171   {
2172     % If not found, try the type specific options.
2173     \bool_lazy_and:nnTF
2174       { ! \tl_if_empty_p:N \l__zrefclever_label_type_a_tl }
2175       {
2176         \prop_if_exist_p:c
2177           { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop }
2178       }
2179       {
2180         \prop_get:cnNF
2181           { l__zrefclever_type_ \l__zrefclever_label_type_a_tl _options_prop } {#1} #2
2182           { \tl_clear:N #2 }
2183       }
2184       { \tl_clear:N #2 }
2185     }
2186   }

```

(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".

```

2187 \cs_new_protected:Npn \__zrefclever_labels_in_sequence:nn #1#2
2188 {
2189   \bool_if:NTF \l__zrefclever_page_ref_bool
2190   {
2191     \exp_args:Nxx \tl_if_eq:nnT
2192       { \zref@extractdefault {#1} { zc@pgfmt } { } }

```

```

2193 { \zref@extractdefault {#2} { zc@pgfmt } { } }
2194 {
2195   \int_compare:nNnTF
2196     { \zref@extractdefault {#1} { zc@pgval } {-2} + 1 }
2197     =
2198     { \zref@extractdefault {#2} { zc@pgval } {-1} }
2199     { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2200     {
2201       \int_compare:nNnT
2202         { \zref@extractdefault {#1} { zc@pgval } {-1} }
2203         =
2204         { \zref@extractdefault {#2} { zc@pgval } {-1} }
2205         {
2206           \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2207           \bool_set_true:N \l__zrefclever_next_is_same_bool
2208         }
2209       }
2210     }
2211   }
2212 {
2213   \exp_args:Nxx \tl_if_eq:nnT
2214   { \zref@extractdefault {#1} { counter } { } }
2215   { \zref@extractdefault {#2} { counter } { } }
2216   {
2217     \exp_args:Nxx \tl_if_eq:nnT
2218     { \zref@extractdefault {#1} { zc@enclval } { } }
2219     { \zref@extractdefault {#2} { zc@enclval } { } }
2220     {
2221       \int_compare:nNnTF
2222         { \zref@extractdefault {#1} { zc@cntval } {-2} + 1 }
2223         =
2224         { \zref@extractdefault {#2} { zc@cntval } {-1} }
2225         { \bool_set_true:N \l__zrefclever_next_maybe_range_bool }
2226         {
2227           \int_compare:nNnT
2228             { \zref@extractdefault {#1} { zc@cntval } {-1} }
2229             =
2230             { \zref@extractdefault {#2} { zc@cntval } {-1} }
2231             {
2232               \bool_set_true:N \l__zrefclever_next_maybe_range_bool
2233               \bool_set_true:N \l__zrefclever_next_is_same_bool
2234             }
2235           }
2236         }
2237       }
2238     }
2239   }

```

(End definition for _zrefclever_labels_in_sequence:nn.)

10 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’.

```
2240 \__zrefclever_add_default_translation:nnn { fallback } { namesep } {\nobreakspace}
2241 \__zrefclever_add_default_translation:nnn { fallback } { pairsep } {,~}
2242 \__zrefclever_add_default_translation:nnn { fallback } { listsep } {,~}
2243 \__zrefclever_add_default_translation:nnn { fallback } { lastsep } {,~}
2244 \__zrefclever_add_default_translation:nnn { fallback } { tpairsep } {,~}
2245 \__zrefclever_add_default_translation:nnn { fallback } { tlistsep } {,~}
2246 \__zrefclever_add_default_translation:nnn { fallback } { tlastsep } {,~}
2247 \__zrefclever_add_default_translation:nnn { fallback } { notesep } {-}
2248 \__zrefclever_add_default_translation:nnn { fallback } { rangeseq } {\textendash}
2249 \__zrefclever_add_default_translation:nnn { fallback } { refpre } {}
2250 \__zrefclever_add_default_translation:nnn { fallback } { refpos } {}
2251 \__zrefclever_add_default_translation:nnn { fallback } { refpre-in } {}
2252 \__zrefclever_add_default_translation:nnn { fallback } { refpos-in } {}

2253 \endpackage
2254 \iflang-english
```

English

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

```
2255 \ProvideDictionaryFor{English}{zref-clever}
2256
2257 \zcdicDefaultTransl{namesep}{\nobreakspace}
2258 \zcdicDefaultTransl{pairsep}{~and\nobreakspace}
2259 \zcdicDefaultTransl{listsep}{,~}
2260 \zcdicDefaultTransl{lastsep}{~and\nobreakspace}
2261 \zcdicDefaultTransl{tpairsep}{~and\nobreakspace}
2262 \zcdicDefaultTransl{tlistsep}{,~}
2263 \zcdicDefaultTransl{tlastsep}{,~and\nobreakspace}
2264 \zcdicDefaultTransl{notesep}{-}
2265 \zcdicDefaultTransl{rangeseq}{~to\nobreakspace}
2266 \zcdicDefaultTransl{refpre}{}
2267 \zcdicDefaultTransl{refpos}{}
2268 \zcdicDefaultTransl{refpre-in}{}
2269 \zcdicDefaultTransl{refpos-in}{}

2270
2271 \zcdicTypeTransl{part}{Name-sg}{Part}
2272 \zcdicTypeTransl{part}{name-sg}{part}
2273 \zcdicTypeTransl{part}{Name-pl}{Parts}
2274 \zcdicTypeTransl{part}{name-pl}{parts}

2275
2276 \zcdicTypeTransl{chapter}{Name-sg}{Chapter}
2277 \zcdicTypeTransl{chapter}{name-sg}{chapter}
2278 \zcdicTypeTransl{chapter}{Name-pl}{Chapters}
```

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

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

```

2387 \zcDicTypeTransl{example}{Name-sg}{Example}
2388 \zcDicTypeTransl{example}{name-sg}{example}
2389 \zcDicTypeTransl{example}{Name-pl}{Examples}
2390 \zcDicTypeTransl{example}{name-pl}{examples}
2391
2392 \zcDicTypeTransl{remark}{Name-sg}{Remark}
2393 \zcDicTypeTransl{remark}{name-sg}{remark}
2394 \zcDicTypeTransl{remark}{Name-pl}{Remarks}
2395 \zcDicTypeTransl{remark}{name-pl}{remarks}
2396
2397 \zcDicTypeTransl{algorithm}{Name-sg}{Algorithm}
2398 \zcDicTypeTransl{algorithm}{name-sg}{algorithm}
2399 \zcDicTypeTransl{algorithm}{Name-pl}{Algorithms}
2400 \zcDicTypeTransl{algorithm}{name-pl}{algorithms}
2401
2402 \zcDicTypeTransl{listing}{Name-sg}{Listing}
2403 \zcDicTypeTransl{listing}{name-sg}{listing}
2404 \zcDicTypeTransl{listing}{Name-pl}{Listings}
2405 \zcDicTypeTransl{listing}{name-pl}{listings}
2406
2407 \zcDicTypeTransl{exercise}{Name-sg}{Exercise}
2408 \zcDicTypeTransl{exercise}{name-sg}{exercise}
2409 \zcDicTypeTransl{exercise}{Name-pl}{Exercises}
2410 \zcDicTypeTransl{exercise}{name-pl}{exercises}
2411
2412 \zcDicTypeTransl{solution}{Name-sg}{Solution}
2413 \zcDicTypeTransl{solution}{name-sg}{solution}
2414 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
2415 \zcDicTypeTransl{solution}{name-pl}{solutions}
2416 </lang-english>
2417 <*lang-german>

```

German

```

2418 \ProvideDictionaryFor{German}{zref-clever}
2419
2420 \zcDicDefaultTransl{namesep}{\nobreakspace}
2421 \zcDicDefaultTransl{pairsep}{~und\nobreakspace}
2422 \zcDicDefaultTransl{listsep}{~,~}
2423 \zcDicDefaultTransl{lastsep}{~und\nobreakspace}
2424 \zcDicDefaultTransl{tpairsep}{~und\nobreakspace}
2425 \zcDicDefaultTransl{tlistsep}{~,~}
2426 \zcDicDefaultTransl{tlastsep}{~und\nobreakspace}
2427 \zcDicDefaultTransl{notesep}{~}
2428 \zcDicDefaultTransl{rangesep}{~bis\nobreakspace}
2429
2430 \zcDicTypeTransl{part}{Name-sg}{Teil}
2431 \zcDicTypeTransl{part}{name-sg}{Teil}
2432 \zcDicTypeTransl{part}{Name-pl}{Teile}
2433 \zcDicTypeTransl{part}{name-pl}{Teile}
2434
2435 \zcDicTypeTransl{chapter}{Name-sg}{Kapitel}
2436 \zcDicTypeTransl{chapter}{name-sg}{Kapitel}

```


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

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

```

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

```

French

```

2567 \ProvideDictionaryFor{French}{zref-clever}
2568
2569 \zcDicDefaultTransl{namesep}{\nobreakspace}
2570 \zcDicDefaultTransl{pairsep}{\et\nobreakspace}
2571 \zcDicDefaultTransl{listsep}{,~}
2572 \zcDicDefaultTransl{lastsep}{\et\nobreakspace}
2573 \zcDicDefaultTransl{tpairsep}{\et\nobreakspace}
2574 \zcDicDefaultTransl{tlistsep}{,~}
2575 \zcDicDefaultTransl{tlastsep}{\et\nobreakspace}
2576 \zcDicDefaultTransl{notesep}{~}
2577 \zcDicDefaultTransl{rangesep}{\à\nobreakspace}
2578
2579 \zcDicTypeTransl{part}{Name-sg}{Partie}
2580 \zcDicTypeTransl{part}{name-sg}{partie}
2581 \zcDicTypeTransl{part}{Name-pl}{Parties}
2582 \zcDicTypeTransl{part}{name-pl}{parties}
2583
2584 \zcDicTypeTransl{chapter}{Name-sg}{Chapitre}
2585 \zcDicTypeTransl{chapter}{name-sg}{chapitre}
2586 \zcDicTypeTransl{chapter}{Name-pl}{Chapitres}
2587 \zcDicTypeTransl{chapter}{name-pl}{chapitres}
2588
2589 \zcDicTypeTransl{section}{Name-sg}{Section}
2590 \zcDicTypeTransl{section}{name-sg}{section}
2591 \zcDicTypeTransl{section}{Name-pl}{Sections}
2592 \zcDicTypeTransl{section}{name-pl}{sections}
2593
2594 \zcDicTypeTransl{paragraph}{Name-sg}{Paragraphe}
2595 \zcDicTypeTransl{paragraph}{name-sg}{paragraphe}

```

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

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

2704 \zcDicTypeTransl{exercise}{name-pl}{exercices}
2705
2706 \zcDicTypeTransl{solution}{Name-sg}{Solution}
2707 \zcDicTypeTransl{solution}{name-sg}{solution}
2708 \zcDicTypeTransl{solution}{Name-pl}{Solutions}
2709 \zcDicTypeTransl{solution}{name-pl}{solutions}
2710 \langle /lang-french \rangle
2711 \langle *lang-portuguese \rangle

Portuguese

2712 \ProvideDictionaryFor{Portuguese}{zref-clever}
2713
2714 \zcDicDefaultTransl{namesep}{\nobreakspace}
2715 \zcDicDefaultTransl{pairsep}{~\nobreakspace}
2716 \zcDicDefaultTransl{listsep}{,~}
2717 \zcDicDefaultTransl{lastsep}{~\nobreakspace}
2718 \zcDicDefaultTransl{tpairsep}{~\nobreakspace}
2719 \zcDicDefaultTransl{tlistsep}{,~}
2720 \zcDicDefaultTransl{tlastsep}{~\nobreakspace}
2721 \zcDicDefaultTransl{notesep}{~}
2722 \zcDicDefaultTransl{rangesep}{~a\nobreakspace}
2723
2724 \zcDicTypeTransl{part}{Name-sg}{Parte}
2725 \zcDicTypeTransl{part}{name-sg}{parte}
2726 \zcDicTypeTransl{part}{Name-pl}{Partes}
2727 \zcDicTypeTransl{part}{name-pl}{partes}
2728
2729 \zcDicTypeTransl{chapter}{Name-sg}{Capítulo}
2730 \zcDicTypeTransl{chapter}{name-sg}{capítulo}
2731 \zcDicTypeTransl{chapter}{Name-pl}{Capítulos}
2732 \zcDicTypeTransl{chapter}{name-pl}{capítulos}
2733
2734 \zcDicTypeTransl{section}{Name-sg}{Seção}
2735 \zcDicTypeTransl{section}{name-sg}{seção}
2736 \zcDicTypeTransl{section}{Name-pl}{Seções}
2737 \zcDicTypeTransl{section}{name-pl}{seções}
2738
2739 \zcDicTypeTransl{paragraph}{Name-sg}{Parágrafo}
2740 \zcDicTypeTransl{paragraph}{name-sg}{parágrafo}
2741 \zcDicTypeTransl{paragraph}{Name-pl}{Parágrafos}
2742 \zcDicTypeTransl{paragraph}{name-pl}{parágrafos}
2743 \zcDicTypeTransl{paragraph}{Name-sg-ab}{Par.}
2744 \zcDicTypeTransl{paragraph}{name-sg-ab}{par.}
2745 \zcDicTypeTransl{paragraph}{Name-pl-ab}{Par.}
2746 \zcDicTypeTransl{paragraph}{name-pl-ab}{par.}
2747
2748 \zcDicTypeTransl{appendix}{Name-sg}{Apêndice}
2749 \zcDicTypeTransl{appendix}{name-sg}{apêndice}
2750 \zcDicTypeTransl{appendix}{Name-pl}{Apêndices}
2751 \zcDicTypeTransl{appendix}{name-pl}{apêndices}
2752
2753 \zcDicTypeTransl{page}{Name-sg}{Página}
2754 \zcDicTypeTransl{page}{name-sg}{página}

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

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


```

2863 \zcDicTypeTransl{exercise}{name-pl}{exercícios}
2864
2865 \zcDicTypeTransl{solution}{Name-sg}{Solução}
2866 \zcDicTypeTransl{solution}{name-sg}{solução}
2867 \zcDicTypeTransl{solution}{Name-pl}{Soluções}
2868 \zcDicTypeTransl{solution}{name-pl}{soluções}
2869
2870 </lang-portuguese>
2871
2872 <*lang-spanish>

```

Spanish

```

2871 \ProvideDictionaryFor{Spanish}{zref-clever}
2872
2873 \zcDicDefaultTransl{namesep}{\nobreakspace}
2874 \zcDicDefaultTransl{pairsep}{~\nobreakspace}
2875 \zcDicDefaultTransl{listsep}{,~}
2876 \zcDicDefaultTransl{lastsep}{~\nobreakspace}
2877 \zcDicDefaultTransl{tpairsep}{~\nobreakspace}
2878 \zcDicDefaultTransl{tlistsep}{,~}
2879 \zcDicDefaultTransl{tlastsep}{~\nobreakspace}
2880 \zcDicDefaultTransl{notesep}{~}
2881 \zcDicDefaultTransl{rangesep}{~a\nobreakspace}
2882
2883 \zcDicTypeTransl{part}{Name-sg}{Parte}
2884 \zcDicTypeTransl{part}{name-sg}{parte}
2885 \zcDicTypeTransl{part}{Name-pl}{Partes}
2886 \zcDicTypeTransl{part}{name-pl}{partes}
2887
2888 \zcDicTypeTransl{chapter}{Name-sg}{Capítulo}
2889 \zcDicTypeTransl{chapter}{name-sg}{capítulo}
2890 \zcDicTypeTransl{chapter}{Name-pl}{Capítulos}
2891 \zcDicTypeTransl{chapter}{name-pl}{capítulos}
2892
2893 \zcDicTypeTransl{section}{Name-sg}{Sección}
2894 \zcDicTypeTransl{section}{name-sg}{sección}
2895 \zcDicTypeTransl{section}{Name-pl}{Secciones}
2896 \zcDicTypeTransl{section}{name-pl}{secciones}
2897
2898 \zcDicTypeTransl{paragraph}{Name-sg}{Párrafo}
2899 \zcDicTypeTransl{paragraph}{name-sg}{párrafo}
2900 \zcDicTypeTransl{paragraph}{Name-pl}{Párrafos}
2901 \zcDicTypeTransl{paragraph}{name-pl}{párrafos}
2902
2903 \zcDicTypeTransl{appendix}{Name-sg}{Apéndice}
2904 \zcDicTypeTransl{appendix}{name-sg}{apéndice}
2905 \zcDicTypeTransl{appendix}{Name-pl}{Apéndices}
2906 \zcDicTypeTransl{appendix}{name-pl}{apéndices}
2907
2908 \zcDicTypeTransl{page}{Name-sg}{Página}
2909 \zcDicTypeTransl{page}{name-sg}{página}
2910 \zcDicTypeTransl{page}{Name-pl}{Páginas}
2911 \zcDicTypeTransl{page}{name-pl}{páginas}
2912
2913 \zcDicTypeTransl{line}{Name-sg}{Línea}

```

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

```

2968 \zcDicTypeTransl{proposition}{name-pl}{proposiciones}
2969
2970 \zcDicTypeTransl{definition}{Name-sg}{Definición}
2971 \zcDicTypeTransl{definition}{name-sg}{definición}
2972 \zcDicTypeTransl{definition}{Name-pl}{Definiciones}
2973 \zcDicTypeTransl{definition}{name-pl}{definiciones}
2974
2975 \zcDicTypeTransl{proof}{Name-sg}{Demostración}
2976 \zcDicTypeTransl{proof}{name-sg}{demostración}
2977 \zcDicTypeTransl{proof}{Name-pl}{Demostraciones}
2978 \zcDicTypeTransl{proof}{name-pl}{demostraciones}
2979
2980 \zcDicTypeTransl{result}{Name-sg}{Resultado}
2981 \zcDicTypeTransl{result}{name-sg}{resultado}
2982 \zcDicTypeTransl{result}{Name-pl}{Resultados}
2983 \zcDicTypeTransl{result}{name-pl}{resultados}
2984
2985 \zcDicTypeTransl{example}{Name-sg}{Ejemplo}
2986 \zcDicTypeTransl{example}{name-sg}{ejemplo}
2987 \zcDicTypeTransl{example}{Name-pl}{Ejemplos}
2988 \zcDicTypeTransl{example}{name-pl}{ejemplos}
2989
2990 \zcDicTypeTransl{remark}{Name-sg}{Observación}
2991 \zcDicTypeTransl{remark}{name-sg}{observación}
2992 \zcDicTypeTransl{remark}{Name-pl}{Observaciones}
2993 \zcDicTypeTransl{remark}{name-pl}{observaciones}
2994
2995 \zcDicTypeTransl{algorithm}{Name-sg}{Algoritmo}
2996 \zcDicTypeTransl{algorithm}{name-sg}{algoritmo}
2997 \zcDicTypeTransl{algorithm}{Name-pl}{Algoritmos}
2998 \zcDicTypeTransl{algorithm}{name-pl}{algoritmos}
2999
3000 \zcDicTypeTransl{listing}{Name-sg}{Listado}
3001 \zcDicTypeTransl{listing}{name-sg}{listado}
3002 \zcDicTypeTransl{listing}{Name-pl}{Listados}
3003 \zcDicTypeTransl{listing}{name-pl}{listados}
3004
3005 \zcDicTypeTransl{exercise}{Name-sg}{Ejercicio}
3006 \zcDicTypeTransl{exercise}{name-sg}{ejercicio}
3007 \zcDicTypeTransl{exercise}{Name-pl}{Ejercicios}
3008 \zcDicTypeTransl{exercise}{name-pl}{ejercicios}
3009
3010 \zcDicTypeTransl{solution}{Name-sg}{Solución}
3011 \zcDicTypeTransl{solution}{name-sg}{solución}
3012 \zcDicTypeTransl{solution}{Name-pl}{Soluciones}
3013 \zcDicTypeTransl{solution}{name-pl}{soluciones}
3014 </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>\AddToHook</code> | |
| . 91, 255, 275, 392, 465, 501, 527, 593 | |
| <code>\addtranslation</code> | 158, 162 |
| <code>\Arg</code> | 2121 |
| <code>\AtEndOfPackage</code> | 15, 499 |
| B | |
| <code>\baselanguage</code> | 490, 551, 567 |
| bool commands: | |
| <code>\bool_case_true:</code> | 2 |
| <code>\bool_if:NTF</code> | |
| . 396, 400, 884, 917, 1284, 1306, | |
| 1379, 1504, 1525, 1556, 1611, 1678, | |
| 1682, 1689, 1698, 1704, 1985, 2189 | |
| <code>\bool_if:nTF</code> ... 59, 946, 955, 964, | |
| 1033, 1061, 1084, 1173, 1181, 1320, | |
| 1328, 1537, 1544, 1551, 1800, 2063 | |
| <code>\bool_lazy_all:nTF</code> | 2128 |
| <code>\bool_lazy_and:nnTF</code> | |
| . 841, 854, 1653, 1862, 2173 | |
| <code>\bool_lazy_any:nTF</code> | 1953, 1962 |
| <code>\bool_lazy_or:nnTF</code> | 845, 1850 |
| <code>\bool_new:N</code> | 239, |
| 300, 301, 326, 353, 362, 369, 370, | |
| 425, 426, 443, 444, 586, 587, 833, | |
| 937, 1213, 1214, 1225, 1226, 1227, 1247 | |
| <code>\bool_set:Nn</code> | 839 |
| <code>\bool_set_false:N</code> .. 246, 260, 265, | |
| 284, 295, 313, 317, 377, 386, 387, | |
| 402, 608, 1022, 1273, 1312, 1326, | |
| 1337, 1516, 1651, 1652, 1960, 1977 | |
| <code>\bool_set_true:N</code> | |
| . 251, 307, 308, 312, 318, 376, 381, | |
| 382, 597, 602, 1045, 1056, 1073, | |
| 1079, 1096, 1102, 1128, 1140, 1280, | |
| 1307, 1313, 1317, 1338, 1341, 1976, | |
| 2199, 2206, 2207, 2225, 2232, 2233 | |
| <code>\bool_until_do:Nn</code> | 1026, 1274 |
| C | |
| clist commands: | |
| <code>\clist_map_inline:Nn</code> | 537, 560 |
| <code>\clist_map_inline:nn</code> | |
| . 202, 620, 666, 682, 746, 771, 801 | |
| <code>\cs</code> 530, 1152, 1502, 2124, 2158, 2169 | |
| cs commands: | |
| <code>\cs:w</code> | 50 |
| <code>\cs_end:</code> | 50 |
| <code>\cs_generate_variant:Nn</code> | |
| . 55, 56, 151, 154, 938, 1841 | |
| <code>\cs_if_exist:NTF</code> | 39, 48, 69 |
| <code>\cs_new:Npn</code> 37, 46, 57, 67, 78, 1796, 1980 | |
| <code>\cs_new_protected:Npn</code> | |
| . 149, 152, 157, 161, 170, 834, 881, | |
| 924, 939, 1003, 1148, 1204, 1250, | |
| 1390, 1646, 1842, 2122, 2167, 2187 | |
| <code>\cs_new_protected:Npx</code> | 90 |
| <code>\cs_set_eq:NN</code> | 94 |
| D | |
| <code>\declaretranslation</code> | 153 |
| <code>\dots</code> | 966, 976, 980 |
| E | |
| <code>\endinput</code> | 12 |
| exp commands: | |
| <code>\exp_args:NNe</code> | 27 |
| <code>\exp_args:NNx</code> | 95, 1070, 1093 |
| <code>\exp_args:Nx</code> 485, 489, 546, 550, 562, 566 | |
| <code>\exp_args:Nxx</code> | |
| . 987, 1041, 2191, 2213, 2217 | |
| <code>\exp_not:N</code> | 103, |
| 109, 1558, 1561, 1581, 1584, 1587, | |
| 1803, 1806, 1809, 1821, 1823, 1826, | |
| 1829, 1834, 1836, 1839, 1983, 1991, | |
| 2009, 2012, 2014, 2017, 2023, 2030, | |
| 2032, 2036, 2039, 2042, 2044, 2050, | |
| 2054, 2057, 2069, 2072, 2075, 2098, | |
| 2100, 2103, 2106, 2112, 2114, 2117 | |
| <code>\exp_not:n</code> . 1410, 1426, 1438, 1442, | |
| 1462, 1475, 1478, 1490, 1493, 1526, | |
| 1527, 1559, 1580, 1585, 1586, 1718, | |
| 1731, 1736, 1756, 1767, 1770, 1780, | |
| 1783, 1804, 1805, 1807, 1817, 1819, | |
| 1822, 1827, 1828, 1830, 1831, 1833, | |
| 1835, 2010, 2011, 2013, 2015, 2016, | |
| 2018, 2019, 2022, 2034, 2035, 2040, | |
| 2041, 2043, 2051, 2055, 2056, 2058, | |
| 2070, 2071, 2073, 2093, 2096, 2099, | |
| 2104, 2105, 2107, 2108, 2111, 2113 | |
| F | |
| file commands: | |
| <code>\file_if_exist:NTF</code> | |
| . 485, 489, 546, 550, 562, 566 | |
| <code>\fmtversion</code> | 3 |

| | | | |
|--------------------|--|---------------------------------------|--|
| G | | M | |
| group commands: | | \MessageBreak | 10 |
| \group_begin: | 93, 836, 1558, 1584, 1803, 1806, 1826, 1829, 2009, 2014, 2017, 2032, 2039, 2054, 2069, 2072, 2103, 2106 | \meta | 1190 |
| \group_end: | 96, 862, 1581, 1587, 1821, 1823, 1834, 1836, 2012, 2023, 2030, 2036, 2042, 2057, 2098, 2100, 2112, 2114 | msg commands: | |
| H | | \msg_line_context: | 102, 108, 117, 131, 135, 137, 139, 141 |
| \hyperlink | 43 | \msg_new:nnn | 100, 106, 111, 113, 115, 120, 125, 127, 129, 134, 136, 138, 140 |
| I | | \msg_warning:nn | 263, 293, 401, 407, 591, 612 |
| \IfBooleanTF | 866 | \msg_warning:nnn | 181, 226, 678, 765, 820, 1354, 1511, 1897, 1944 |
| \IfFormatAtLeastTF | 3, 4 | \msg_warning:nnnn | 1054, 1077, 1100, 1138 |
| \IfTranslation | 144 | N | |
| int commands: | | \NewDocumentCommand | 164, 167, 655, 659, 727, 830, 864 |
| \int_case:nnTF | 1393, 1419, 1450, 1614, 1711, 1747 | \NewHook | 460 |
| \int_compare:nNnTF | 991, 1046, 1113, 1129, 1159, 1161, 1206, 1361, 1406, 1440, 1603, 1605, 1668, 1692, 1734, 2195, 2201, 2221, 2227 | \nobreakspace | 2240, 2257, 2258, 2260, 2261, 2263, 2265, 2420, 2421, 2423, 2424, 2426, 2428, 2569, 2570, 2572, 2573, 2575, 2577, 2714, 2715, 2717, 2718, 2720, 2722, 2873, 2874, 2876, 2877, 2879, 2881 |
| \int_compare_p:nNn | 1175, 1183, 1854, 1865, 1973 | \noexpand | 123 |
| \int_eval:n | 90 | P | |
| \int_incr:N | 1641, 1681, 1683, 1697, 1699, 1703, 1705, 1794 | \PackageError | 7 |
| \int_new:N | 870, 871, 1220, 1221, 1222, 1223 | \pagenumbering | 5 |
| \int_set:Nn | 1160, 1162, 1166, 1169 | \pkg | 529, 539, 542 |
| \int_use:N | 33, 35, 50 | prg commands: | |
| \int_zero:N | 1150, 1151, 1258, 1259, 1260, 1261, 1640, 1642, 1643, 1789, 1790 | \prg_generate_conditional_variant:Nnn | 148 |
| iow commands: | | \prg_new_conditional:Npnn | 142 |
| \iow_newline: | 122, 126 | \prg_return_false: | 146 |
| K | | \prg_return_true: | 145 |
| keys commands: | | \ProcessKeysOptions | 15, 654 |
| \keys_define:nn | 19, 176, 198, 221, 240, 279, 289, 302, 327, 336, 354, 363, 371, 404, 411, 427, 445, 461, 503, 581, 588, 598, 609, 617, 642, 674, 708, 733, 754, 784, 813 | prop commands: | |
| \keys_set:nn | 19, 603, 656, 664, 731, 837 | \prop_get:NnN | 2141 |
| keyval commands: | | \prop_get:NnNTF | 1876, 1903, 1908, 2125, 2170, 2180 |
| \keyval_parse:nnn | 180, 225 | \prop_if_exist:NTF | 661, 740 |
| L | | \prop_if_exist_p:N | 2132, 2176 |
| \labelformat | 2 | \prop_if_in:NnTF | 25 |
| \LoadDictionaryFor | 487, 491, 548, 552, 564, 568, 574 | \prop_if_in_p:Nn | 60, 2136 |
| | | \prop_item:Nn | 27, 61 |
| | | \prop_new:N | 169, 220, 619, 662, 741 |
| | | \prop_put:Nnn | 174, 649, 720 |
| | | \prop_remove:Nn | 173, 648, 715 |
| | | \providecommand | 3 |
| | | \ProvideDictionaryFor | 2255, 2418, 2567, 2712, 2871 |
| | | \ProvideDictTranslation | 165, 168 |
| | | \ProvidesExplPackage | 14 |

| | |
|---|---|
| R | |
| <code>\refstepcounter</code> | 2 |
| <code>\RequirePackage</code> 16, 17, 18, 19, 20, 397, 653 | |
| S | |
| <code>\SaveTranslationFor</code> | 150 |
| seq commands: | |
| <code>\seq_clear:N</code> | 350, 883 |
| <code>\seq_get_left:NN</code> | 1282 |
| <code>\seq_if_empty:N</code> | 1277 |
| <code>\seq_if_in:NnTF</code> | 204, 930 |
| <code>\seq_map_break:n</code> | 81, 1195, 1198 |
| <code>\seq_map_function:NN</code> | 886 |
| <code>\seq_map_indexed_inline:Nn</code> ... | 1155 |
| <code>\seq_map_inline:Nn</code> | 1192 |
| <code>\seq_map_tokens:Nn</code> | 63 |
| <code>\seq_new:N</code> ... | 197, 335, 832, 880, 1215 |
| <code>\seq_pop_left:NN</code> | 1276 |
| <code>\seq_put_right:Nn</code> | 205, 932 |
| <code>\seq_reverse:N</code> | 344 |
| <code>\seq_set_eq:NN</code> | 1252 |
| <code>\seq_set_from_clist:Nn</code> | 340, 838 |
| <code>\seq_sort:Nn</code> | 889 |
| <code>\setcounter</code> | 5 |
| sort commands: | |
| <code>\sort_return_same:</code> | |
| | 25, 31, 896, 901, 953, |
| | 996, 998, 1051, 1057, 1074, 1080, |
| | 1103, 1134, 1141, 1179, 1195, 1211 |
| <code>\sort_return_swapped:</code> | |
| | 25, 31, 909, 962, |
| | 995, 1050, 1097, 1133, 1187, 1198, 1210 |
| str commands: | |
| <code>\str_case:nnTF</code> | 467, 507 |
| <code>\str_if_eq:nnTF</code> | 80 |
| <code>\str_if_eq_p:nn</code> | 1958, 1964, 1966, 1970 |
| <code>\str_new:N</code> | 410 |
| <code>\str_set:Nn</code> | 415, 417, 419, 421 |
| T | |
| T _E X and L ^A T _E X 2 _ε commands: | |
| <code>\@currentcounter</code> .. | 21, 25, 28, 33, 84, 86 |
| <code>\@currentlabel</code> | 2 |
| <code>\@ifl@t@r</code> | 3 |
| <code>\@ifpackageloaded</code> | |
| | 257, 277, 394, 534, 557, 595 |
| <code>\@trns@lt@current@language</code> | 531 |
| <code>\@trns@lt@language</code> | 486, 547, 563 |
| <code>\bbl@loaded</code> | 537 |
| <code>\bbl@main@language</code> | 536 |
| <code>\c@</code> | 3 |
| <code>\c@page</code> | 5, 94 |
| <code>\cl@</code> | 3 |
| <code>\hyper@link</code> .. | 1561, 1809, 1991, 2075 |
| <code>\p@...</code> | 2 |
| <code>\xpg@loaded</code> | 560 |
| <code>\xpg@main@language</code> | 559 |
| <code>\zref@addprop</code> .. | 22, 32, 34, 36, 87, 88, 99 |
| <code>\zref@default</code> | |
| | 43, 1839, 1983, 2044, 2050, 2117 |
| <code>\zref@extractdefault</code> | |
| | 43, 927, 942, 944, 988, |
| | 989, 992, 994, 1006, 1010, 1014, |
| | 1018, 1042, 1043, 1047, 1049, 1069, |
| | 1092, 1207, 1209, 1292, 1297, 1566, |
| | 1571, 1577, 1812, 1813, 1815, 1818, |
| | 1832, 1996, 2000, 2005, 2020, 2080, |
| | 2084, 2089, 2094, 2109, 2192, 2193, |
| | 2196, 2198, 2202, 2204, 2214, 2215, |
| | 2218, 2219, 2222, 2224, 2228, 2230 |
| <code>\zref@ifpropundefined</code> | 10 |
| <code>\zref@ifrefcontainsprop</code> .. | 10, 1563, |
| | 1798, 1811, 1987, 1993, 2060, 2077 |
| <code>\zref@ifrefundefined</code> | |
| .. | 891, 893, 905, 1309, 1311, 1316, |
| | 1349, 1508, 1517, 1659, 1844, 1982 |
| <code>\ZREF@mainlist</code> .. | 22, 32, 34, 36, 87, 88, 99 |
| <code>\zref@newprop</code> .. | 4, 21, 23, 33, 35, 83, 85, 98 |
| <code>\zref@refused</code> | 1348 |
| <code>\zref@wrapper@babel</code> | 22, 831 |
| <code>\textendash</code> | 2248 |
| <code>\texttt</code> | 850 |
| <code>\the</code> | 2 |
| <code>\thepage</code> | 5, 95 |
| tl commands: | |
| <code>\c_empty_tl</code> | 927, 942, |
| | 944, 1006, 1010, 1014, 1018, 1293, 1298 |
| <code>\c_novalue_tl</code> | 644, 710 |
| <code>\tl_clear:N</code> .. | 730, 738, 1024, 1025, |
| | 1253, 1254, 1255, 1256, 1257, 1279, |
| | 1636, 1637, 1638, 1639, 1680, 1845, |
| | 1848, 1872, 1896, 1943, 2182, 2184 |
| <code>\tl_gset:Nn</code> | 95 |
| <code>\tl_head:N</code> | |
| .. | 1029, 1031, 1114, 1116, 1130, 1132 |
| <code>\tl_if_empty:N</code> | 71, 759, 789, |
| | 818, 928, 1352, 1506, 1859, 1874, 2048 |
| <code>\tl_if_empty:nTF</code> | |
| | 172, 737, 1458, 1473, |
| | 1488, 1729, 1754, 1765, 1778, 1847 |
| <code>\tl_if_empty_p:N</code> ... | 950, 951, 959, |
| | 960, 967, 968, 1323, 1324, 1331, |
| | 1332, 1957, 1967, 1971, 2130, 2174 |
| <code>\tl_if_empty_p:n</code> .. | 1037, 1038, 1065, 1088 |
| <code>\tl_if_eq:NNTF</code> | 974, 1111, 1335 |
| <code>\tl_if_eq:NnTF</code> .. | 1165, 1168, 1194, 1197 |
| <code>\tl_if_eq:nnTF</code> | |
| ... | 987, 1041, 1157, 2191, 2213, 2217 |

| | |
|---|---|
| 2736, 2737, 2739, 2740, 2741, 2742, 2743, 2744, 2745, 2746, 2748, 2749, 2750, 2751, 2753, 2754, 2755, 2756, 2757, 2758, 2760, 2761, 2762, 2763, 2765, 2766, 2767, 2768, 2769, 2770, 2771, 2772, 2774, 2775, 2776, 2777, 2779, 2780, 2781, 2782, 2784, 2785, 2786, 2787, 2789, 2790, 2791, 2792, 2794, 2795, 2796, 2797, 2798, 2799, 2800, 2801, 2802, 2803, 2805, 2806, 2807, 2808, 2810, 2811, 2812, 2813, 2815, 2816, 2817, 2818, 2820, 2821, 2822, 2823, 2825, 2826, 2827, 2828, 2830, 2831, 2832, 2833, 2835, 2836, 2837, 2838, 2840, 2841, 2842, 2843, 2845, 2846, 2847, 2848, 2850, 2851, 2852, 2853, 2855, 2856, 2857, 2858, 2860, 2861, 2862, 2863, 2865, 2866, 2867, 2868, 2883, 2884, 2885, 2886, 2888, 2889, 2890, 2891, 2893, 2894, 2895, 2896, 2898, 2899, 2900, 2901, 2903, 2904, 2905, 2906, 2908, 2909, 2910, 2911, 2913, 2914, 2915, 2916, 2918, 2919, 2920, 2921, 2923, 2924, 2925, 2926, 2928, 2929, 2930, 2931, 2933, 2934, 2935, 2936, 2938, 2939, 2940, 2941, 2943, 2944, 2945, 2946, 2947, 2948, 2950, 2951, 2952, 2953, 2955, 2956, 2957, 2958, 2960, 2961, 2962, 2963, 2965, 2966, 2967, 2968, 2970, 2971, 2972, 2973, 2975, 2976, 2977, 2978, 2980, 2981, 2982, 2983, 2985, 2986, 2987, 2988, 2990, 2991, 2992, 2993, 2995, 2996, 2997, 2998, 3000, 3001, 3002, 3003, 3005, 3006, 3007, 3008, 3010, 3011, 3012, 3013 | __zrefclever_add_type_translation:nnnn 161 |
| \zcheck | \l__zrefclever_capitalize_bool 425, 429, 1851 |
| \zcpageref | \l__zrefclever_capitalize_first_- bool 426, 435, 1853 |
| \zcRefTypeSetup | __zrefclever_counter_reset_by:n 4, 39, 41, 43, 48, 50, 52, 57 |
| \zcsetup | __zrefclever_counter_reset_by_- aux:nn 57 |
| zrefcheck commands: | __zrefclever_counter_reset_by_- auxi:nnn 57 |
| \zrefcheck_zcRef_beg_label: ... | \l__zrefclever_counter_resetby_- prop 60, 61, 220, 227 |
| \zrefcheck_zcRef_end_label_- maybe: | \l__zrefclever_counter_resettters_- seq 3, 63, 197, 204, 205 |
| \zrefcheck_zcRef_run_checks_on_- labels:n | \l__zrefclever_counter_type_prop 3, 25, 28, 169, 182 |
| zrefclever internal commands: | \l__zrefclever_current_language_- tl 459, 478, 518, 531 |
| \l__zrefclever_abbrev_bool | __zrefclever_declare_translation:nnn 152, 154, 761, 791, 795, 824 |
| 443, 447, 1863 | __zrefclever_get_enclosing_- counters:n 4, 37, 42, 84 |
| __zrefclever_add_default_- translation:nnn ... 157, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252 | __zrefclever_get_enclosing_- counters_value:n ... 4, 37, 51, 86 |
| | __zrefclever_get_option_- plain:nN ... 1363, 1364, 1365, 2167 |
| | __zrefclever_get_option_with_- transl:nN .. 52, 53, 1264, 1265, 1266, 1267, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 2121 |
| | __zrefclever_get_ref:n 1411, 1427, 1439, 1443, 1463, 1476, 1479, 1491, 1494, 1528, 1548, 1719, 1732, 1737, 1757, 1768, 1771, 1781, 1784, 1796 |
| | __zrefclever_get_ref_first: 33, 43, 1541, 1597, 1980 |
| | __zrefclever_get_translation_- for:nnn 149, 151, 1888, 1920, 1935, 2150, 2160 |
| | __zrefclever_if_translation:nn 142, 148 |
| | __zrefclever_if_translation:nnTF 1881, 1913, 1928, 2146 |
| | \l__zrefclever_label_a_tl 872, 1276, 1293, 1309, 1348, 1349, 1355, 1399, 1411, 1427, 1443, 1479, 1494, 1521, 1528, 1659, 1663, 1672, 1696, 1719, 1737, 1771, 1784 |
| | \l__zrefclever_label_b_tl 872, 1279, 1282, 1298, 1311, 1316, 1663 |
| | \l__zrefclever_label_count_int 32, 1220, |

1258, 1361, 1393, 1640, 1668, 1794

\l_zrefclever_label_enclcnt_a-
tl 872, 1005,
1007, 1008, 1029, 1094, 1118, 1119

\l_zrefclever_label_enclcnt_b-
tl 872, 1009,
1011, 1012, 1031, 1071, 1120, 1121

\l_zrefclever_label_enclval_a-
tl 872, 1013,
1015, 1016, 1114, 1122, 1123, 1130

\l_zrefclever_label_enclval_b-
tl 872, 1017,
1019, 1020, 1116, 1124, 1125, 1132

\l_zrefclever_label_type_a_tl ..
..... 872, 926, 928, 930,
933, 941, 950, 959, 967, 974, 1165,
1194, 1286, 1290, 1323, 1331, 1336,
1352, 1400, 1673, 2130, 2133, 2137,
2142, 2148, 2152, 2174, 2177, 2181

\l_zrefclever_label_type_b_tl ..
872, 943, 951, 960, 968, 974, 1168,
1197, 1287, 1295, 1324, 1332, 1336

_zrefclever_label_type_put_-
new_right:n 25, 887, 924

\l_zrefclever_label_types_seq ..
..... 25, 880, 883, 930, 933, 1192

_zrefclever_labels_in_sequence:nn
..... 1520, 1662, 2187

\l_zrefclever_last_of_type_bool
..... 31, 1213, 1307, 1312, 1313,
1317, 1326, 1337, 1338, 1341, 1379

\l_zrefclever_lastsep_tl . 1236,
1370, 1426, 1442, 1462, 1478, 1490

\l_zrefclever_link_star_bool ...
..... 832, 839, 1801, 1956, 2066

\l_zrefclever_listsep_tl
... 1235, 1369, 1438, 1475, 1718,
1731, 1736, 1756, 1767, 1770, 1780

\l_zrefclever_main_language_tl .
..... 458, 472, 512, 536, 559, 573

\l_zrefclever_name_format_-
fallback_tl 1246,
1869, 1872, 1874, 1910, 1932, 1939

\l_zrefclever_name_format_tl ...
..... 1246,
1856, 1857, 1860, 1861, 1869, 1870,
1878, 1885, 1892, 1905, 1917, 1924

\l_zrefclever_name_in_link_bool
.. 1246, 1556, 1960, 1976, 1977, 1985

\l_zrefclever_namefont_tl 1228,
1363, 1559, 1585, 2010, 2040, 2055

\l_zrefclever_nameinlink_str ...
..... 410, 415,
417, 419, 421, 1958, 1964, 1966, 1970

\l_zrefclever_nameinlink_tl .. 410

\l_zrefclever_namesep_tl
.. 1232, 1366, 2013, 2043, 2051, 2058

\l_zrefclever_next_is_same_bool
..... 32, 51, 1222,
1652, 1682, 1698, 1704, 2207, 2233

\l_zrefclever_next_maybe_range_-
bool
.. 32, 51, 1222, 1516, 1525, 1651,
1678, 1689, 2199, 2206, 2225, 2232

\l_zrefclever_noabbrev_first_-
bool 444, 453, 1866

\l_zrefclever_notesept_tl
..... 851, 1241, 1267

_zrefclever_page_format_aux: ..
..... 90, 94

\g_zrefclever_page_format_tl ...
..... 5, 89, 95, 98

_zrefclever_page_numbering: ... 89

\l_zrefclever_page_ref_bool ...
..... 239, 246, 251,
260, 265, 284, 295, 884, 917, 1284, 2189

\l_zrefclever_pairsep_tl
..... 1234, 1368, 1410, 1526

_zrefclever_prop_put_non_-
empty:Nnn 170, 182, 227

\l_zrefclever_range_beg_label_-
tl 32, 1222, 1257,
1439, 1458, 1463, 1473, 1476, 1488,
1491, 1639, 1680, 1696, 1729, 1732,
1754, 1757, 1765, 1768, 1778, 1781

\l_zrefclever_range_count_int ..
..... 32,
1222, 1260, 1419, 1451, 1642, 1681,
1693, 1697, 1703, 1711, 1748, 1789

\l_zrefclever_range_inhibit_-
next_bool 31, 32, 1222, 1657

\l_zrefclever_range_same_count_-
int 32,
1222, 1261, 1406, 1440, 1451, 1643,
1683, 1699, 1705, 1734, 1748, 1790

\l_zrefclever_rangesep_tl
..... 1233, 1367, 1493, 1527, 1783

\l_zrefclever_ref_language_tl ..
..... 457, 472, 478, 482,
512, 518, 521, 1882, 1889, 1914,
1921, 1929, 1936, 2147, 2151, 2161

\l_zrefclever_ref_options_prop .
..... 19, 619, 648, 649, 2125, 2170

\l_zrefclever_ref_property_tl ..
..... 10, 238, 245,
250, 259, 264, 283, 294, 1798, 1818,
1832, 1988, 2021, 2061, 2095, 2110

`\l_zrefclever_ref_typeset_font_-`
`tl` 616, 618, 1270
`\l_zrefclever_reffont_in_tl` 1230,
1365, 1807, 1830, 2018, 2073, 2107
`\l_zrefclever_reffont_out_tl` ...
..... 1229, 1364,
1804, 1827, 2015, 2034, 2070, 2104
`\l_zrefclever_refpos_in_tl` 1245,
1374, 1819, 1833, 2022, 2096, 2111
`\l_zrefclever_refpos_out_tl` 1243,
1372, 1822, 1835, 2035, 2099, 2113
`\l_zrefclever_refpre_in_tl` 1244,
1373, 1817, 1831, 2019, 2093, 2108
`\l_zrefclever_refpre_out_tl` 1242,
1371, 1805, 1828, 2016, 2071, 2105
`\l_zrefclever_setup_language_tl`
..... 657, 729, 761, 791, 795, 824
`\l_zrefclever_setup_type_tl` ...
..... 657, 663, 716, 721,
730, 738, 742, 759, 789, 796, 818, 825
`\l_zrefclever_sort_decided_bool`
.... 937, 1022, 1026, 1045, 1056,
1073, 1079, 1096, 1102, 1128, 1140
`_zrefclever_sort_default:nn` ...
..... 24–26, 919, 939
`_zrefclever_sort_default_-`
`different_types:nn` 981, 1148
`_zrefclever_sort_default_same_-`
`type:nn` 977, 1003
`_zrefclever_sort_labels:`
..... 25, 31, 848, 881
`_zrefclever_sort_page:nn`
..... 31, 918, 1204
`\l_zrefclever_sort_prior_a_int` .
870, 1150, 1159, 1160, 1166, 1176, 1184
`\l_zrefclever_sort_prior_b_int` .
871, 1151, 1161, 1162, 1169, 1177, 1185
`\l_zrefclever_tlastsep_tl`
..... 1240, 1266, 1628
`\l_zrefclever_tlistsep_tl`
..... 1239, 1265, 1606
`\l_zrefclever_tpairsep_tl`
..... 1238, 1264, 1622
`\l_zrefclever_type_<type>-`
`options_prop` 19
`\l_zrefclever_type_count_int` ...
..... 32, 1220, 1259, 1603,
1605, 1614, 1641, 1854, 1865, 1973
`\l_zrefclever_type_first_label_-`
`tl` 1215, 1255, 1399, 1508,
1517, 1521, 1548, 1564, 1567, 1572,
1578, 1637, 1672, 1844, 1982, 1988,
1994, 1996, 2000, 2005, 2020, 2061,
2078, 2080, 2084, 2089, 2094, 2109
`\l_zrefclever_type_first_label_-`
`type_tl` 1215,
1256, 1400, 1512, 1638, 1673, 1847,
1877, 1884, 1891, 1898, 1904,
1909, 1916, 1923, 1931, 1938, 1945
`_zrefclever_type_name_setup:` ..
..... 33, 1536, 1842
`\l_zrefclever_type_name_tl` . 44,
1246, 1580, 1586, 1845, 1848, 1879,
1888, 1896, 1906, 1911, 1920, 1935,
1943, 1957, 2011, 2041, 2048, 2056
`\l_zrefclever_typeset_compress_-`
`bool` 353, 356, 1654
`\l_zrefclever_typeset_labels_-`
`seq` ... 1215, 1252, 1276, 1277, 1282
`\l_zrefclever_typeset_label_bool`
..... 31, 1213,
1273, 1274, 1280, 1306, 1611, 1972
`\l_zrefclever_typeset_name_bool`
..... 301, 308, 313, 318, 1538, 1552
`\l_zrefclever_typeset_queue_-`
`curr_tl` 1215, 1254, 1408,
1424, 1433, 1460, 1470, 1485, 1506,
1523, 1540, 1547, 1554, 1597, 1618,
1623, 1629, 1635, 1636, 1716, 1727,
1752, 1763, 1776, 1859, 1967, 1971
`\l_zrefclever_typeset_queue_-`
`prev_tl` 1215, 1253, 1607, 1635
`\l_zrefclever_typeset_range_-`
`bool` 362, 365, 847, 1504
`\l_zrefclever_typeset_ref_bool` .
..... 300, 307, 312, 317, 1538, 1545
`_zrefclever_typeset_refs:`
..... 31, 32, 43, 44, 47, 849, 1250
`_zrefclever_typeset_refs_aux_-`
`last_of_type:` 1382, 1390
`_zrefclever_typeset_refs_aux_-`
`not_last_of_type:` 1386, 1646
`\l_zrefclever_typeset_sort_bool`
..... 326, 329, 846
`\l_zrefclever_typesort_seq`
..... 335, 340, 344, 350, 1155
`\l_zrefclever_use_hyperref_bool`
..... 369, 396, 402, 1801, 1955, 2065
`\l_zrefclever_warn_hyperref_-`
`bool` 369, 400
`_zrefclever_zcref:nnn` 831, 834
`_zrefclever_zcref:nnnn` . 22, 24, 834
`\l_zrefclever_zcref_labels_seq` .
.... 24, 832, 838, 860, 887, 889, 1252
`\l_zrefclever_zcref_note_tl` ...
..... 580, 583, 852
`\l_zrefclever_zcref_with_check_-`
`bool` 587, 602, 843, 856

| | | |
|----------------------------|-------|-------------------------|
| \l__zrefclever_zrefcheck_- | | 586, 597, 608, 842, 855 |
| available_bool | | |