enumitem-zref

Extended references to items for enumitem package

The \refitem command

FC

2011/02/18 – version 1.8

Abstract

enumitem-zref, as a companion package for enumitem[1], extends the references mechanism for lists of items and allows hyperlinks to any items in lists formatted by enumitem. Hyper-references to lists seems not always necessary, but their importance rise much when a document contains lists that can stretch out on several pages.

Primarily intented for **enumerate** environments (*ie.* numbered lists), enumitem-zref can also reference any item in **description** or **itemize** environments: restrictions can be set with the package options.

A natural reference scheme for referencing items anywhere in the document is provided with three modalities (the *commaref* scheme, the *refitem* scheme and the *user* scheme). The references can be typeset differently with the command \refitem , its options and eventually arbitrary text.

Additionnally, enumitem-zref allows to typeset greek-enumerated lists with the greek or greekctr package options, which use code from packages alphalph[2] and engrec[3] or greekctr[4].

References to items in external documents compiled with enumitem-zref is also possible with pdfTEX[5].

enumitem-zref requires and is based on the packages enumitem by Javier Bezos and zref[6] by Heiko Oberdiek. It works with an ε-TEX distribution of LATEX.

Contents

1. enumitem: CTAN:help/Catalogue/entries/enumitem.html
2. alphalph: CTAN:help/Catalogue/entries/alphalph.html
3. engrec: CTAN:help/Catalogue/entries/engrec.html
4. greekctr: CTAN:help/Catalogue/entries/greekctr.html
5. This is not xr but "pure" PDF.
6. zref: CTAN:help/Catalogue/entries/zref.html

This documentation is produced with the DocStrip utility.

⟶ To get the package,	run:	etex enumitem-zref.dtx
⟶ To get the documentation	run (thrice):	pdflatex enumitem-zref.dtx
To get the index,	run:	makeindex -s gind.ist enumitem-zref.idx

The .dtx file is embedded into this pdf file thank to embedfile by H. Oberdiek.

Summary of the properties for \refitem

Terms definitions

\label {user} This is a user *label*.

\item [user] This is a user *item*.

\item This is an *automatic* item.

User items are labelized, hence expanded (with \protect).

The \label command can be put inside the optional argument of \item:

\item [text \label {label of the item}]

or in the paragraph that follows.

Package hyperref's [verbose] option reports all inserted anchor names in the .log file.

Syntax for \refitem

\refitem {reference} [properties to display]

Optional text containing properties
Default is [\ref]

reference with either:
- The *refitem* scheme: listname{label}{label}{label}...
- The *commaref* scheme: listname,index,index,index,...
- The *user* scheme: reference to a \label command

The order in which the properties are given is related to the depth of the list:

[\ref \ref \ref]

- \ref: refers to the second ancestor *Here an empty string*
- \ref: refers to the parent 1.
- \ref: refers to the item ii)

1. One
   i) sub item
   ii) sub item
2. Two

	Property	Description	Examples
most common properties	\label	The item label: automatic or user defined by \item [...]	II) or •
	\label*	For automatic labels in enumerate lists: the label without any text around (useful to "concatenate" labels). Otherwise, the same as \label.	II or •
	\ref	If the ref key of enumitem has been set, the "evaluation" of the ref key. Otherwise, the same as \label	II) or •
	\ref*	The same as \label*	II or •
	\listname	The name given to the list with the key name.	
	\depth	The list depth.	
	\index	The index of the item in the list. Every item is counted.	
	\listctr	The value of \@listctr for the item – only inside enumerate lists. May be different from \index if the list contains user *items*.	
	\type	The type of the list: either enumerate or itemize or description.	
	\currentlabel	LaTeX's \@currentlabel	
hyperref	\anchor	The name of the anchor set for the item.	
hyperref	\currentlabelname	hyperref's (or nameref's) \@currentlabelname	
	\refitem	The list of references with the *refitem* scheme for the item.	\zeninfo
	\commaref	The list of references with the *commaref* scheme for the item.	\zeninfo
	\default	The same as \ref	

1 Forewords

1.1 Numbered lists, not numbered lists and hyperref

LaTeX defines two "classes" of lists of items:

- **enumerate** lists which are *numbered* (*ie.* a counter gives an index to each item in the list),
- **description** and **itemize** lists which are *not numbered* (no counter, thus no index).

When hyperref is used in *numbered lists*, each \item gives hyperref the opportunity to define an anchor, whose name is chosen considering the current value of the *index*.

Different cases have to be considered:

1) We are in a *numbered* list (*ie.* enumerate):
   i) if the item is automatic (\item without option), hyperref puts an anchor named Item.IDX.
   ii) if the item is user defined (\item [user item]), hyperref does not put an anchor.
2) We are not in a *numbered* list (*ie.* description or itemize):
   hyperref does not put an anchor.

Moreover, when hyperref is loaded, the expansion of the \label command leads to the expansion of the item label (with \protect) for it is written to the .aux file:

- in the case 1 i)
- and also in the case 1 ii) if:
  - the list is a description list, and
  - package nameref is loaded, and
  - the \label command appears inside the user item text.

I was guardedly affraid about those rules when I understood them, as you may be...

enumitem-zref adds an anchor in the cases 1 ii) and 2). The item label text is always expanded (with \protect and babel protection).

1.2 A "natural" reference scheme for lists

enumitem-zref provides **3 ways** to reference a specific item in a list. The first 2 can be used to reference items in lists of external documents as well.

References are made with the command: \refitem , but \hyperref can be used as well.

\subitem and \subsubitem provided in some classes cannot be referenced with enumitem-zref.

1.2.1 The *commaref* scheme

Assuming you gave a name to your enumerate, description or itemize list, and let's say this name is 'myList', each item is given an index. Any item is indexed, whether it's printed by the mean of a user item or an an automatic item. Then:

- this is the item myList,1
- this is the item myList,2
  - this is the item myList,2,1
  - this is the item myList,2,2

user-defined this is the item myList,2,3

Reference to the third item can be made with:

\refitem {myList,2,1}	this prints: 1 for it is the first item in the inner list.
\refitem {myList,2,1}[\ref.\ref]	prints: 2.1.
\refitem {myList,2,1}[\label \ \label]	prints: • –.

Nothing simplest, no more comment...

1.2.2 The *refitem* scheme

A) this is the item myList{A}.

B) this is the item myList{B} (as you'd have expected...)

i) this is the item myList{B}{i}.
and is *also* innerlist{i} if this nested list has been named 'innerlist'.

ii) this is the item myList{B}{ii} (or innerlist{ii}).

α) this is the item myList{B}{ii}{\alpha} (with the package option [greek]).

β) this is the item myList{B}{ii}{\beta} (or innerlist{ii}{\beta}).

C) this is the item myList{C} (as you'd have expected...)

perso this is the item myList{perso}.

D) this is the item myList{D} and is also myList,5 in the *commaref* scheme.

- this is the item myList{D}{1}
- this is the item myList{D}{2}
inside itemize lists, labels are replaced by their arabic index in the *refitem* scheme.

And here is a reference to:

\refitem {myList{B}{ii}{\alpha }}	α
\refitem {myList{B}{ii}{\alpha }}[\label \ \label \ \label]	B) ii) α)
\refitem {myList{B}{ii}{\alpha }}[\label∗--\label∗--\label]	B–ii–α)

Now what happens with a description list:

1°) this is the item myList{1}

2°) this is the item myList{2}

First description label this is the item myList{2}{first description label}

And a second one this is the item myList{2}{and a second one}

3°) this is the item myList{3}

\refitem {myList{2}{first description label}}	First description label
\refitem {myList2{2}{first description label}}[\ref∗, \label]	2, First description label
\refitem {myList2{2}{first description label}}[\ref \ \ref]	2°) First description label

The correspondance between the reference name and the reference that is finally printed is immediate: the *refitem* scheme is almost useful when used with hyperref.

The list name myList is case sensitive, but the item number {B} or {first description label} is not. This can be changed with the CaSeS package option. Trailing spaces in the item label are always removed from the reference name in the *refitem* scheme.

1.2.3 The *user* scheme: with the \label command

1– this is the item myList,1 and myList{1} and as long as I type \label {Important Item} it is also named Important Item.

2– this is the item myList,2 and myList{2}.

a) this is the item myList,2,1 and myList{2}{a}.

b) this is the item myList,2,2 and myList{2}{b} and as long as I type: \label {Important subitem} it is also named Important subitem.

Then you get:

\refitem {Important subitem}	b)
\refitem {Important subitem}[\label \ \label]	2– b)
\refitem {Important subitem}[\ref∗ sub point \ref]	2 sub point b)

The reference is case sentitive.

2 User Interface

2.1 Naming lists

To use the reference schemes defined by enumitem-zref, it is necessary to give lists a name. For that purpose, two new keys: `name` and `name*` have been added to the keys provided by enumitem.

The key 'name' gives a name to the list, while `name*` is a bit more clever and should be used in every circonstances when lists are nested, almost when lists are automatically named by the mean of the paragraph they belong to. Example:

```
\newlist{myList}{enumerate}{3}
\setlist[myList]{ label* = \arabic*., name* = \arabic{section}.\arabic{subsection}}
```

In case of nested lists, `name*` will compare the name you gave to the list with the name of its parent list, and ignore the key in case of equality.

This gives:

1. First item
2. Second item
   2.1. sub item
   2.2. another sub item
   2.3. a last sub item `\label {for further reference}`

And then:

`\refitem {2.1,2,3}`	2.3.	in the *commaref* scheme
`\refitem {2.1{2}{3}}`	2.3.	in the *refitem* scheme
`\refitem {for further reference}`	2.3.	in the *user* scheme

To name your lists automatically it is also possible to use (these are examples):

- interfaces-hypbmsec with package hypbmsec:
  `name* = \csname the\lastsectionname \endcsname`
- package refcount: `name* = \getrefbykeydefault {reference}{anchor}{default}`.

2.2 Referencing the items : `\refitem`

`\refitem {⟨reference⟩}[text with properties \ref, \label etc..]`[babel]
`\refitem *`

enumitem-zref provides `\refitem` to reference the items of lists with the following possible syntaxes:

`\refitem {refname}[properties]`	displays a reference to `refname` the text displayed depends on the optional properties whose default value is `\ref`.
`\refitem * {refname}[properties]`	can be used when you want a reference but no hyperlink.

If the reference `rename` does not point to an item, `\refitem` behaves like `\ref` (or `\ref*`) and the `[properties]` are ignored. A warning is displayed in the .log file.

The properties available with `\refitem` are:

`\ref` This is the default property: the same as LaTeX's `\ref` command: it expands to the text corresponding to the `ref` key of the enumitem list environment, which by default is:

- in `enumerate` lists: the same as the `label`.
- in `itemize` lists: the index of the item.
- in `description` lists: the user defined label, or if no label has been given to the `\item` command, the index of the item.

\label	The label is printed, whatever value the ref key has been given. For description, the label may be empty.
\label* \ref*	This is a variant of \label: automatic labels in enumerate lists are related to a counter. \ref* or \label* prints the formatted value of the counter, without any text around. For itemize, description or user items inside enumerate lists, this is the same as \label.
\page	The same as \pageref.
\name	The name that has been given to the list.
\depth	The depth of the list (equal to 1 if the list is not nested).
\index	The index of the item; user defined items are counted.
\listctr	The item number inside numbered lists (*ie.* enumerate); user defined items are not counted: this is the value of the counter \@listctr.
\type	The type of the list (*ie.* enumerate, itemize or description)
\refitem	The list of references names that can be invoked with \refitem for this specific item in the *refitem* scheme (see also \zeninfo).
\commaref	The list of references names that can be invoked with \refitem for this specific item in the *commaref* scheme (see also \zeninfo).

\refitem [file:...]{⟨*reference*⟩}{⟨*text*⟩}$^{\text{babel}}$

For reference to external files (to make hyperlinks), \refitem can be invoked with the following syntax:

\refitem [file:...]{refname}{text}	Can be used for references to an external .pdf file.
\refitem []{refname}{text}	The current file is used by default...

Only the *refitem* and the *commaref* schemes can be used. The user scheme cannot be used presently for external files. An implementation "à la xr" might be provided in a further release.

2.3 enumitem-zref package options

enumitem-zref can be loaded with the following options:

defaults:

true	enumerate	disables the references for itemize and description lists: only enumerated lists can be referenced.
true	itemize	enables/disables the references for itemize lists.
true	description	enables/disables the references for description lists.
false	greek	Defines \engrec (lowercase) and \EnGrec (uppercase) to make enumerate lists of items with the greek alphabet as the label (package engrec). \engrec and \EnGrec can be used in place of \arabic (f.ex. label=\engrec *).
false	greekctr	Defines \greek and \Greek (like greek option but with package greekctr[7]).
false	CaSeS	Activate this option to get CaSeS sEnSiTiV references in the *refitem* scheme. This option applies only to the refitem scheme. List names are always case sensitive. If this option is enabled with hyperref, anchors corresponding to the *refitem* scheme will be case sensitiv as well. Package hyperref's [verbose] option allows to see anchor names in the .log file.

	CaSeS = true	CaSeS = false
	I) First	I) First
	II) Second	II) Second
\refitem {myList{I}}	✔works	✔works
\refitem {myList{i}}	✗doesn't work	✔works
\refitem {mylist{I}}	✗doesn't work	✗doesn't work

List names are always case sensitive.

7. greekctr: CTAN:help/Catalogue/entries/greekctr.html

Types of lists that can be referenced			
Package option	*enumerate*	*itemize*	*description*
[enumerate]	•		
[itemize]	•	•	
[description]	•		•
[enumerate,itemize]	•	•	
[enumerate,description]	•		•
[itemize,description]	•	•	•
[enumerate,itemize,description]	•	•	•
none of the above	•	•	•

2.4 If your are lost \zeninfo is your friend

\zeninfo [info-key]

In the paragraph following an item, \zeninfo reports some informations about the references to the current item:

depth The depth of the list.
index The index of the item.
commaref The references that can be used to reference the item in the *commaref* scheme.
refitem The reference that can be used to reference the item in the *refitem* scheme.
anchor The anchor set for the item (if hyperref).

With no argument, \zeninfo prints all these informations.

Example with the pifont package:

```
\newcommand\@Ding[1]{\ding{\number\numexpr#1 + 171}}
\newcommand\Ding[1]{\@Ding{\value{#1}}}
\AddEnumerateCounter \Ding \@Ding {}

\begin{enumerate}[label=\Ding*,name=dingList]
\item One
\item Two
\item Three                \zeninfo
\end{enumerate}
```

① One
② Two
③ Three zeninfo:depth=1/ idx=3/ ctr=3/ commaref={dingList,3}/ refitem={dingList{\protect \fontfamily {pzd}\protect \fontencoding {u}\protect \fontseries {m}\protect \fontshape {n}\protect \selectfont \par@update \char 174}}/ anchor=Item.37

Now:

\refitem {dingList{\@Ding {3}}} produces ③

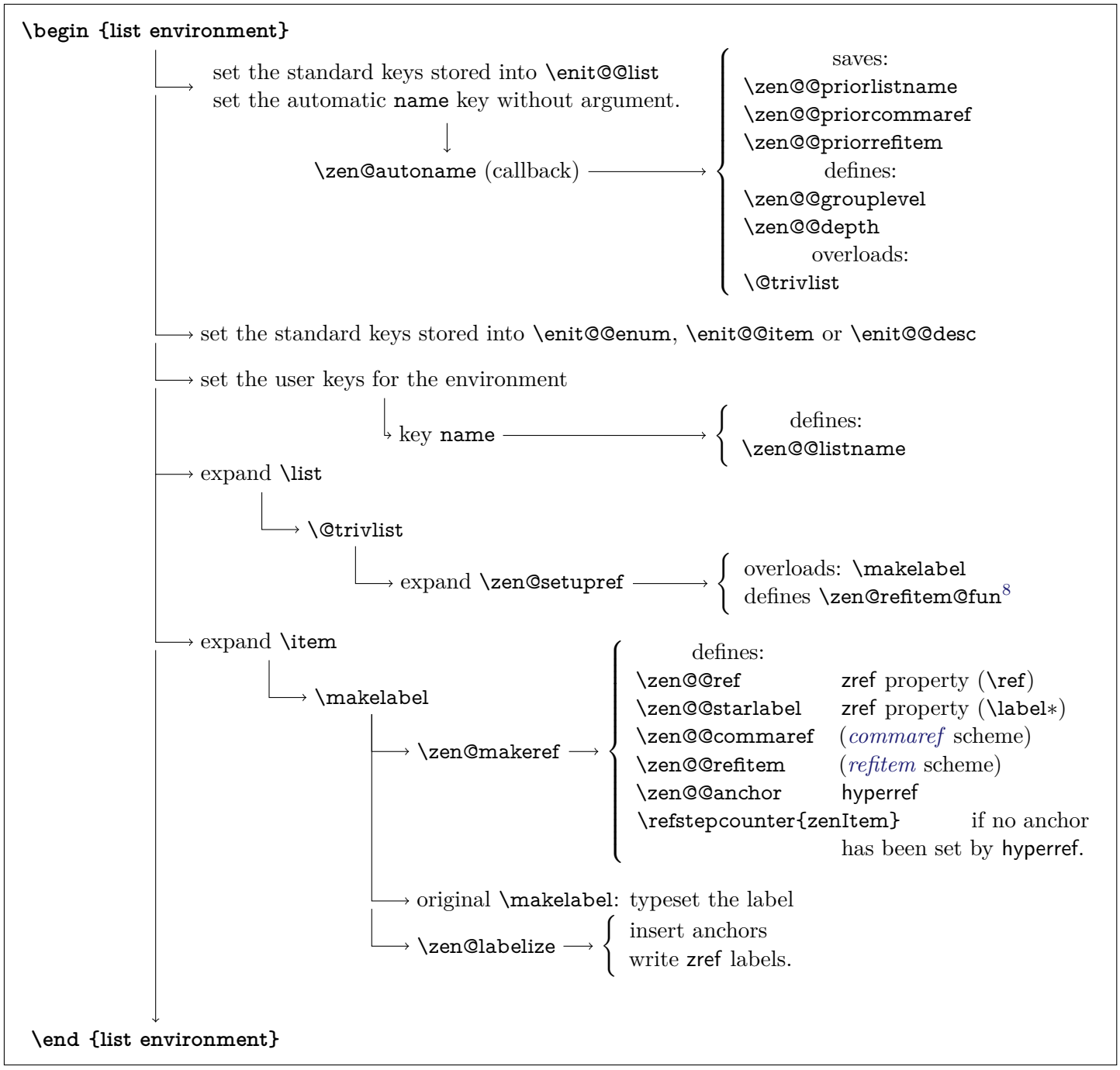
Note that for this purpose:

- \@Ding could be named differently (without @).
- the *commaref* scheme is easier: \refitem {dingList,3} gives ③.

This example shows the output of \zeninfo.

7. For the French language with babel frenchb-setup, itemize lists can be processed by enumitem-zref only if the option:
\frenchbsetup {CompactItemize=false}
has been set. This option is automatically set by enumitem-zref when option [itemize] is passed.

3 Flow chart of expansion: How does it work ?



8. Used for the property \label* and to define the label (and the anchor) in the *refitem* scheme.

4 Implementation

4.1 Identification and requirements

The package namespace is **\zen@**

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[2005/12/01]
\ProvidesPackage{enumitem-zref}[2011/02/18 v1.8 - Extended references for enumitem pkg]
\RequirePackage{ltxcmds,etoolbox,kvoptions,gettitlestring,enumitem}
\RequirePackage{zref}[2010/05/01]
\RequirePackage{zref-user,zref-counter}
\GetTitleStringDisableCommands{\let\enit@format \@empty}
```

Some \catcode assertions:

```
\let\zen@AtEnd \@empty
\def\TMP@EnsureCode#1#2{{%
  \count@\catcode#1\relax
  \toks@\expandafter{\zen@AtEnd \catcode#1 }
  \xdef\zen@AtEnd{\the\toks@ \the\count@\relax}}
  \catcode#1 #2\relax
}
\TMP@EnsureCode{40}{12}    % (
\TMP@EnsureCode{41}{12}    % )
\TMP@EnsureCode{`\Ï}{14}   % Ï (default comment: itemize option)
\TMP@EnsureCode{`\Ð}{14}   % Ð (default comment: description option)
\TMP@EnsureCode{`\Ë}{9}    % Ë (default ignore: external option)
```

4.2 The package options

```
\SetupKeyvalOptions{family=zen,prefix=zen@}
\DeclareBoolOption{CaSeS}
\DeclareBoolOption{itemize}
\DeclareBoolOption{description}
\DeclareBoolOption{enumerate}
\DeclareVoidOption{greek}{\AtEndOfPackage\zen@option@greek}
\def\zen@option@greek{%
  \RequirePackage{engrec,alphalph}
  \let\zen@grecORI \@grec \let\zen@GrecORI \@Grec
  \let\@grec \relax       \let\@Grec \relax
  \newalphalph\@grec[wrap]\zen@grecORI{24}
  \newalphalph\@Grec[wrap]\zen@GrecORI{24}
  \AddEnumerateCounter\engrec\@grec{\@grec{23}}% widest is \psi
  \AddEnumerateCounter\EnGrec\@Grec{\@Grec{12}}% widest is M
  \let\zen@option@greek \relax
}% \zen@option@greek
\DeclareVoidOption{greekctr}{\AtEndOfPackage\zen@option@greekctr}
\def\zen@option@greekctr{%
  \RequirePackage{greekctr,alphalph}
  \let\zen@greekORI \@greek \let\zen@GreekORI \@Greek
  \let\@greek \relax       \let\@Greek \relax
  \newalphalph\@greek[wrap]\zen@greekORI{24}%
  \newalphalph\@Greek[wrap]\zen@GreekORI{24}%
  \AddEnumerateCounter\greek\@greek{\@greek{23}}% widest is \psi
  \AddEnumerateCounter\Greek\@Greek{\@Greek{12}}% widest is M
  \let\zen@option@greekctr \relax
}% \zen@option@greekctr
\DeclareVoidOption{external}{\catcode`\Ë 14\relax}% Ë = comment
\DeclareDefaultOption{\@unknownoptionerror}
\AtBeginDocument{%
  \@ifpackageloaded{greekctr} \zen@option@greekctr \relax
  \@ifpackageloaded{engrec} \zen@option@greek      \relax
}
\ProcessLocalKeyvalOptions{zen}
```

```
\ifzen@enumerate\else
  \ifzen@itemize     \zen@enumeratetrue\else
  \ifzen@description \zen@enumeratetrue\else
  \zen@enumeratetrue \zen@itemizetrue \zen@descriptiontrue
  \fi\fi
  \ifzen@enumerate\else
    \zen@enumeratetrue \PackageWarning{enumitem-zref}
         {Option 'enumerate = false' has no effect}%
  \fi
\fi
\ifzen@itemize    \catcode`\Ï 9  \fi   % ignore Ï
\ifzen@description\catcode`\Ð 9  \fi   % ignore Ð
\AtBeginDocument{\zen@item@comma \setcounter{zenItem}\z@}
```

Babel French: frenchb redefines \itemize at begin document.

Therefore, when option [itemize] of enumitem-zref is used, one have to ensure that babel-french setup CompactItemize=false is set:

```
Ï \AtEndPreamble{\ifdefined\frenchbsetup\frenchbsetup{CompactItemize=false}\fi}% [itemize] option
```

4.3 Some constants

zenItem is the global LaTeX counter for zref labels and anchors added by enumitem-zref

\c@zenidx is the local counter for the index of the items. Every item is counted.

\zen@toks A token to be used by \refitem at the time the (optional) properties are analyzed: we can't use \toks@ at this time, because \toks@ could have been set to a value anywhere else.

```
\newcounter{zenItem}
\globcount\c@zenidx
\newtoks\zen@toks
```

\zen@@autolabel To check if the item user-defined or automatic.

```
\def\zen@@autolabel{\@itemlabel}
```

\zen@@noreference How to display undefined references.

```
\def\zen@@noreference{\nfss@text{\reset@font\bfseries ??}}
```

\zen@Hy@anchor is used to put the anchors for the items. It's a wrapper for hyperref \Hy@raisedlink.

```
\protected\def\zen@Hy@anchor#1{%
  \Hy@raisedlink{\hyper@anchorstart{#1}\hyper@anchorend}%
}% \zen@Hy@anchor (wrapper for \Hy@raisedlink)
```

\zen@lowercase \zen@CaSeS is used to make lowercase references in the *refitem* scheme, when the package option CaSeS is not active.

```
\let\zen@CaSeS \@firstofone
\ifzen@CaSeS   \let\zen@lowercase \zen@CaSeS
\else          \let\zen@lowercase \lowercase
\fi
```

\zen@ifrefundefined Switch to the first part if the LaTeX label exists, to the second part otherwise.

```
\def\zen@ifrefundefined#1{\ltx@IfUndefined{r@#1}}
```

\zen@box A utility to expand code inside a temporary \vbox

```
\def\zen@box{\hfuzz\maxdimen \vfuzz\hfuzz \hbadness\@M \vbadness\hbadness
             \tracinglostchars\z@ \everypar{}\setbox\z@=\vbox}
```

4.4 Properties for the zref list: zen@list

\zen@unique is the unique counter to create unique labels for zref references (internal use only):

```
\zref@newlist{zen@list}
\def\zen@{zen>\the\c@zenItem}
\zref@newprop{zen@}[]\zen@
\def\zen@temp#1#2#3{%
  \zref@newprop{zen@#2}[\zen@@noreference]{#3}%
  \ifx U#1\expandafter\let\csname zen@extract@#2\endcsname\unexpanded
  \else  \expandafter\let\csname zen@extract@#2\endcsname\detokenize
  \fi
}%
\zref@newprop*{zen@page}[\zen@@noreference]\thepage
\expandafter\let\csname zen@extract@page\endcsname\unexpanded
\zen@temp U{anchor}\zen@@anchor
\zen@temp U{default}\zen@@ref
\zen@temp U{type}{\ifdefined\zen@@type\ifnum\zen@@type<\z@\else \enit@type \fi\fi}
\zen@temp D{listname}\zen@@listname
\zen@temp U{depth}{\the\@listdepth}
\zen@temp U{index}{\the\c@zenidx}
\zen@temp U{listctr}{\ifdefined\@listctr \the\value\@listctr\fi}
\zen@temp U{parents}\zen@@parents
\zen@temp U{label}\zen@@itemlabel
\zen@temp U{starlabel}\zen@@starlabel
\zen@temp U{ref}\zen@@ref
\zen@temp U{item}\@itemlabel
\zen@temp U{currentlabel}\@currentlabel
\zen@temp U{currentlabelname}\@currentlabelname
\zen@temp D{refitem}\zen@@refitem
\zen@temp D{commaref}\zen@@commaref
\zen@temp U{counter}{\zref@getcurrent{counter}}
\zref@addprop{zen@list}{%
  zen@default,zen@page,zen@type,zen@depth,zen@listctr,%
  zen@listname,zen@index,zen@label,zen@ref,zen@starlabel,%
  zen@parents,zen@refitem,zen@commaref,zen@item,zen@currentlabel,zen@counter}
\newif\ifzen@Hy
\AtBeginDocument{%
  \ifdefined\hyper@anchor \zen@Hytrue
    \zref@addprop{zen@list}{zen@anchor,zen@currentlabelname}%
  \else \zen@Hyfalse  \let\zen@refHy \zen@refText
  \fi
}
```

4.5 Normalisation of references and item labels

\zen@Normalize

```
\def\zen@Normalize#1#2#3{\begingroup \let\zen@=#1%
  \def\GetTitleStringResult{#3}\toks@{}%
  \ifx \GetTitleStringResult\@empty \else
    \zen@NormalizeCommands    \GetTitleStringExpand{#3}%
    \expandafter\zen@trailingspaces \GetTitleStringResult{\@nnil}\@nnil
  \fi
  \edef\x{\endgroup
    \def\noexpand#2{\expandafter\zen@zap@doublespaces
                \detokenize\expandafter{\GetTitleStringResult} \@empty}%
    \def\noexpand\zen@temp {\expandafter\zen@zap@doublespaces
                \detokenize\expandafter{\the\toks@} \@empty}%
  }\x
}% \zen@Normalize
\def\zen@trailingspaces#1#{\toks@{#1}\expandafter\zen@trailingsp@ces}
{\catcode47 = 8 % /
\gdef\zen@trailingsp@ces#1{%
  \ifx \@nnil#1\@empty \expandafter\remove@to@nnil
```

```
  \else \zen@{\toks@\expandafter{\the\expandafter\toks@
                  \expandafter{\romannumeral0\zen@postspace #1/ /}}}%
      \expandafter\zen@trailingsp@ces       % (loop)
  \fi
}% \zen@trailingsp@ces
\gdef\zen@postspace#1 /{\zen@postsp@ce#1/}
\gdef\zen@postsp@ce#1/#2{ #1}
}% \catcode group
\def\zen@NormalizeCommands{%
  \let\ensuremath \@firstofone        % \ensuremath (engrec package)
  \let\relax \@empty
  \csname @safe@activestrue\endcsname
}% \zen@NormalizeCommands
\def\zen@zap@doublespaces#1 #2{%
  #1%
  \ifx#2\@empty\else\space\expandafter\zen@zap@doublespaces\fi
  #2%
}% \zen@zap@doublespaces
```

4.6 Keys name and name∗ are added to enumitem environments

```
\define@key{enumitem}{name}[]{\zen@namelist{#1}}
\define@key{enumitem}{name*}[]{\zen@starnamelist{#1}}
```

\zen@namelist is the callback for the key name.

```
Ë \AtBeginDocument{\gappto\enit@@list{,name}}% ignore if [external]
\def\zen@namelist #1{%
  \ifdefined\zen@@grouplevel\else \let\zen@@grouplevel \z@ \fi
  \ifnum\zen@@grouplevel<\currentgrouplevel% first (automatic) assignment  to name
    \zen@type                          % = (automatic from \enit@@list)
    \ifcase\zen@@type           % (case 0) = enum
      \zen@autoname
    \or                       % (case 1) = itemize
Ï     \zen@autoname             % [itemize] option only
    \or                       % (case 2) = description
Ð     \zen@autoname             % [description] option only
    \fi
  \else \zen@Normalize\zen@CaSeS\zen@@listname {#1}%
      \let\zen@@listname \zen@temp
  \fi
}% \zen@namelist (code for key name)
```

\zen@starnamelist is the callback for the key name∗.

```
\def\zen@starnamelist #1{% (1st automatic assignment to name done before)
  \zen@Normalize\zen@CaSeS\zen@@listname{#1}%
  \let\zen@@listname \zen@temp
  \ifx \zen@@priorlistname\zen@@listname
      \let\zen@@listname \@empty \fi
  \zen@namelist\zen@@listname
}% \zen@starnamelist (code for key name*)
```

\zen@autoname is the callback for the key name automatically set without argument by \setkeys {enit@@list}. This corresponds to some setup:

- save the prior list name *etc.*.
- define the group level,
- define the list depth,
- reset the index,
- save the definition of \@trivlist and overload it temporarily until its next expansion at the beginning of the list.

```
\def\zen@autoname{% (1st automatic assignment to key name (by \enit@@list))
  \unless \ifx\zen@@listname\@empty
```

```
    \let\zen@@priorlistname \zen@@listname \fi
  \unless\ifdefined\zen@@priorlistname \let\zen@@priorlistname \@empty\fi
  \let\zen@@listname \@empty
  \ifdefined\zen@@commaref \let\zen@@priorcommaref \zen@@commaref
  \else                    \let\zen@@priorcommaref \@empty
                           \let\zen@@commaref      \@empty
  \fi
  \ifdefined\zen@@refitem \let\zen@@priorrefitem \zen@@refitem
  \else                   \let\zen@@priorrefitem \@empty
                          \let\zen@@refitem      \@empty
  \fi
  \edef\zen@@grouplevel {\the\currentgrouplevel}%
  \edef\zen@@depth {\number\@listdepth}%
  \ifnum \zen@@depth>\@ne \edef\zen@@parents{{\the\c@zenItem}\zen@@parents}%
  \else                   \let\zen@@parents \@empty \fi
  \let\zen@refitem@fun \@undefined
  \ifcase \zen@@type \zen@prepare@resume \fi   % (enumerate only: key resume)
  \c@zenidx \z@
  \let\zen@trivlistORI \@trivlist
  \let\@trivlist \zen@trivlist
}% \zen@name@list
\def\zen@prepare@resume{\expandafter\def\expandafter\enit@endenumerate
  \expandafter{\enit@endenumerate \zen@endenumerate}%
}% \zen@prepare@resume
\def\zen@endenumerate{% saves the value of \c@zenidx at the exit of enum envir
  \zen@toks\expandafter{\enit@afterlist}%
  \xdef\enit@afterlist{\the\zen@toks
     \def\expandafter\noexpand
       \csname enit@resume@\@currenvir\endcsname{%
       \csname c@\@listctr\endcsname
       \the\csname c@\@listctr\endcsname
       \c@zenidx \the\c@zenidx\relax}}%
}% \zen@prepare@resume
```

\zen@trivlist `\@trivlist` is temporarily overloaded, and the original definition is immediately restored at the time it will be expanded, just after having set all the specified keys for the list environment.

The overload sets `\zen@setuprefs` to be expanded at the end of `\@trivlist`.

```
\def\zen@trivlist{%
  \let\@trivlist \zen@trivlistORI     % restore orginal immediately
  \@trivlist     \zen@setuprefs
}% \zen@trivlist
```

\zen@type expands to:

- 0 if **enumerate**
- 1 if **itemize** (and **[itemize]** option is active)
- 2 if **description** (and **[description]** option is active)
- -1 in any other case

```
\def\zen@type{%
  \ifdefined\enit@type
    \expandafter\strip@prefix       \if e\enit@type>\let\zen@@type\z@  % enum begins with e
    \else>\expandafter\strip@prefix  \if i\enit@type>\let\zen@@type\@ne % item begins with i
    \else>\expandafter\strip@prefix  \if d\enit@type>\let\zen@@type\tw@ % desc begins with d
    \else>\let\zen@@type\m@ne
    \fi\fi\fi
  \else    \let\zen@@type\m@ne
  \fi
}% \zen@type
```

4.7 Overloading \makelabel

\zen@setuprefs expands just at the end of \@trivlist. \makelabel is overloaded in order to make the references and write the labels into the .aux file.

\zen@makelabel

```
\def\zen@setuprefs{%
  \edef \zen@temp{\zen@@priorlistname\zen@@listname}%
  \ifx \zen@temp\@empty
    \ifx \makelabel\zen@makelabel \let\makelabel \zen@makelabelORI \fi
  \else
    \ifx \makelabel\zen@makelabel \else
      \let \zen@makelabelORI \makelabel
      \let \makelabel \zen@makelabel   \fi
  \fi
  \let\zeninfo \zen@info
  \ifcase \zen@@type        % enumerate only (key start)
    \ifnum \c@zenidx=\z@ \c@zenidx\value\@enumctr\relax \fi
  \fi
  \zen@refitem@implicit
}% \zen@setuprefs
\def\zen@makelabel#1{\zen@makeref{#1}\zen@makelabelORI{#1}\zen@labelize}
```

\zen@refitem@implicit defines \zen@refitem@fun which cancels text around the formatted counter in enumerate lists (*ie.* numbered lists). This will be used at the time of \makelabel to create the refitem and the property \label∗ (which corresponds to zref property zen@starlabel).

```
\def\zen@refitem@implicit{% determine value of key refitem according to the label
  \def\zen@gtemp {{\the\c@zenidx}}%
  \ifnum \zen@@type=\z@      % enumerate
    {\let\enit@refstar \zen@refitemfromlabel
     \enit@labellist   \zen@box{\@itemlabel}}%
  \fi
  \let\zen@refitem@fun \zen@gtemp
}% \zen@refitem@implicit
\def\zen@refitemfromlabel#1{\def#1##1{0%
  \def\zen@temp{##1}\ifx \zen@temp\@enumctr
                \gdef\zen@gtemp{{#1{##1}}}\fi}%
}% \zen@refitemfromlabel
```

4.8 Creating references

\zen@makeref is expanded just before the original version of \makelabel

The aim is to set the zref properties defined by enumitem-zref for the current item, and to define the anchor names.

```
\def\zen@makeref#1{% #1 = argument of \makelabel
  \def\zen@@itemlabel{#1}\advance\c@zenidx \@ne
  \ifx \zen@@itemlabel\zen@@autolabel
    \if@nmbrlist
      \global\advance\c@zenItem \@ne
      \let\zen@@ref \@currentlabel
    \else
      {\refstepcounter{zenItem}}%
      \let\zen@@ref \zen@refitem@fun
    \fi
    \let\zen@@starlabel \zen@refitem@fun
    \zen@Normalize\zen@lowercase\zen@currentrefitem \zen@refitem@fun
  \else                          % user defined label
    {\refstepcounter{zenItem}}%
    \zen@Normalize\zen@lowercase\zen@@ref {{\zen@@itemlabel}}%
    \let\zen@@starlabel \zen@@ref
  \fi
  \let\zen@currentrefitem \zen@temp     % \zen@temp defined by \zen@Normalize
```

```
277   \let\zen@@anchor \@currentHref
278   \zen@makeref@item@comma
279 }% \zen@makeref
```

\zen@labelize is expanded just after the original version of \makelabel

zlabels are written into the .aux file.

```
280 \def\zen@labelize{%
281   \ifzen@Hy \zen@anchors
282           \zref@labelbyprops{zen->\@currentHref}{zen@}\fi
283   \zref@labelbylist\zen@{zen@list}%
284   \zen@keeplabel
285 }% \zen@makeref
```

\zen@makeref@item@comma Creates the references for the *commaref* and the *refitem* schemes.

```
286 \def\zen@makeref@item@comma{%
287 \begingroup    % to avoid silent assignment to \relax
288   \edef\x{\endgroup
289     \def\noexpand\zen@@commaref{%
290       \expandafter\zen@commaref@\zen@@priorcommaref \@nnil}%
291     \def\noexpand\zen@@refitem{%
292       \expandafter\zen@refitem@\zen@@priorrefitem \@nnil}%
293   }\x
294 }% \zen@makeref@item@comma
295 \def\zen@refitem@#1{%
296   \ifx\@nnil#1%
297     \ifx \zen@@listname\@empty \else
298       {\zen@@listname\zen@currentrefitem}\fi
299   \else {#1\zen@currentrefitem}\expandafter\zen@refitem@ % loop
300   \fi
301 }% \zen@refitem@
302 \def\zen@commaref@#1{%
303   \ifx\@nnil#1%
304     \ifx \zen@@listname\@empty \else
305       {\zen@@listname,\the\c@zenidx}\fi
306   \else {#1,\the\c@zenidx}\expandafter\zen@commaref@     % loop
307   \fi
308 }% \zen@commaref@
```

\zen@anchors Put the PDFanchors for the current item.

```
309 \let\zen@list@anchors\@empty
310 \def\zen@anchors{\begingroup
311   \def\zen@@anchors{\endgroup}%
312   \expandafter\expandafter\expandafter\zen@anchors@
313       \expandafter\zen@@commaref \zen@@refitem \@nnil
314   \zen@@anchors
315 }% \zen@anchors
316 \def\zen@anchors@#1{%
317   \ifx \@nnil#1\else
318     \expandafter\in@\csname refitem.#1\expandafter\endcsname\expandafter{\zen@list@anchors}%
319     \ifin@   \zen@anchor@ignore{#1}%
320     \else
321       \edef\zen@@anchors{\zen@@anchors\zen@Hy@anchor{refitem.#1}}%
322       \xdef\zen@list@anchors{%
323         \unexpanded\expandafter{\zen@list@anchors}%
324         \expandafter\noexpand\csname refitem.#1\endcsname}%
325     \fi
326     \expandafter\zen@anchors@   % loop
327   \fi
328 }% \zen@anchors@ (loop)
329 \def\zen@anchor@ignore#1{\PackageInfo{enumitem-zref}{Ignored anchor '#1'}}
```

\zen@keeplabel As far as references are built and written inside \makelabel (which was overloaded) they are built inside a group (inside a \sbox, and thus inside a \color@setgroup...\color@endgroup group if packages

color or xcolor are used, and may be inside other grouping levels depending on the user need (she may have redefined \makelabel for a special purpose).

Therefore, we take advantage of the e-TEX command \currentgrouplevel in order to decide how many levels of grouping we have to "pass through" for keeping the current values of \z@zenidx (the item *index*), \zen@@refitem, and \zen@@commaref.

The "target" group level has been saved into \zen@@grouplevel at the beginning of the environment, while setting the automatic name key for the first time.

```
\def\zen@keep#1{\def\noexpand#1{\unexpanded\expandafter{#1}}}%
\def\zen@keeplabel{%
  \xdef\zen@gtemp{%
    \noexpand\ifnum \zen@@grouplevel<\currentgrouplevel
      \aftergroup \noexpand\zen@gtemp
    \noexpand\else
      \c@zenidx \the\c@zenidx\relax
      \zen@keep \zen@@refitem
      \zen@keep \zen@@commaref
      \ifzen@Hy \zen@keep\zen@@anchor\fi
    \noexpand\fi
  }\aftergroup\zen@gtemp
}% \zen@keeplabel
```

4.9 Extracting zref properties

\zen@extract A wrapper to \zref@extractdefault which can also \detokenize the result.

```
\newcommand*\zen@extract[3]{% #1=ref #2=prop #3=def
  \csname zen@extract@#2\endcsname\expandafter\expandafter%
    \expandafter{\zref@extractdefault{zen>\number#1}{zen@#2}{#3}}%
}% \zen@extract
```

4.10 Referencing the items: \refitem

\refitem The general macro to print references and insert hyperlinks, with a star variant (no hyperlink).

```
\newrobustcmd*\refitem{\begingroup \@ifstar
  {\zen@refitemOpt \zen@reftext }%        no hyperlink
  {\zen@refitemOpt \zen@refHy }%
}% \refitem
\def\zen@refHy{%
%%  \edef\zen@@anchor{\zen@extract\c@zenItem{anchor}{}\unexpanded}%
%%  \ifx \zen@@anchor\@empty \zen@@text
%%  \else
  \hyperref{}{}{\zref@extract{zen>\the\c@zenItem}{zen@anchor}}\zen@@text
%%  \fi
}% \zen@ref
\def\zen@refText{\zen@@text}
```

\zen@refitemOpt If the first argument is an option, then \refitem makes a hyperlink to an external document.

```
\def\zen@refitemOpt#1{\let\zen@ref@#1\@ifnextchar[\zen@refitemExternal \zen@refitemRef }
\def\zen@refitemRef #1{\@tempswatrue \let\zen@lastwarn \m@ne
  \zen@Normalize\zen@lowercase\zen@@label {#1}%
  \zen@ifrefundefined \zen@@label
    {\expandafter\expandafter\expandafter \zen@refused
      \csname\ifcsname zen@refitem(\zen@temp)\endcsname
                  zen@refitem(\zen@temp)\else
                  zen@norefitem\fi\endcsname>}%
    \zen@refitemFromAnchor
}% \zen@refitemRef
\def\zen@norefitem {\z@>0}
```

\zen@refused Checks if the label exists in the *commaref* scheme or in the *refitem* scheme.

```
\def\zen@refused#1>#2>{%
```

```
\ifcase#1\relax
    \@tempswafalse    \refused\zen@@label
    \let\zen@ref@ \zen@refText % do not create a hyperlink if no reference
\or   \let\zen@@label \zen@temp \c@zenItem =#2\relax
\else \let\zen@@label \zen@temp \zen@MultipleLabels
    \c@zenItem =#2\relax
\fi
\zen@refitem@prop
}% \zen@refused
```

`\zen@refitemFromAnchor` When `\refitem` refers to a user item, we can reached the zref properties with the chain:

LaTeX label ⟶ hyperref anchor ⟶ enumitem-zref label ⟶ property

```
\def\zen@refitemFromAnchor{%
  \edef\zen@@anchor{\getrefbykeydefault\zen@@label{anchor}{}}%
  \ifx \zen@@anchor\@empty
    \@tempswafalse \G@refundefinedtrue \zen@warn\z@
    \let\zen@ref@ \zen@refText
  \else
    \edef\zen@temp{\zref@extractdefault{zen->\zen@@anchor}{zen@}{}}%
    \ifx \zen@temp\@empty
      \@tempswafalse \G@refundefinedtrue \zen@warn\@ne
      \ifzen@Hy \ifx\zen@ref@ \zen@refHy \def\zen@ref@{\ref{\zen@@label}}%
              \else \def\zen@ref@ {\ref*{\zen@@label}}%
              \fi
      \else \def\zen@ref@{\ref{\zen@@label}}%
      \fi
    \else \c@zenItem \expandafter\strip@prefix\zen@temp\relax
    \fi
  \fi
  \zen@refitem@prop
}% \zen@refitemFromAnchor
```

```
\def\zen@warn#1{\let\zen@lastwarn#1%
  \message{Package enumitem-zref Warning:
  \ifcase#1 %
    \string\refitem{\zen@@label}:
    User references to \string\label\space^^J\zen@spaces
    require hyperref and the appropriate options for enumitem-zref^^J\zen@spaces
    \ifzen@Hy Check the options given to package enumitem-zref.
    \else Package hyperref is not loaded.\fi
  \or User reference does not point to an item!^^J\zen@spaces
     \string\refitem{\zen@@label}
  \fi\on@line^^J^^J}}
\edef\zen@spaces{\@spaces\@spaces}
```

`\zen@refitem@prop` tests if `\refitem` has a following optional argument: the properties.

As long as the optional argument is in last position, we have to take a special care about the the spaces (`\@ifnextchar` removes spaces).

```
\def\zen@refitem@prop{\futurelet\zen@temp \zen@refitemProp}
\def\zen@gobblespace#1 {#1}%
\let\zen@space \relax
\def\zen@refitemProp{%
  \ifcase 0\ifx \zen@temp[\else
          \ifx \zen@temp\@sptoken 1\else
          2\fi\fi\relax
       \expandafter \zen@refitemPrint
  \or   \let\zen@space\space
       \expandafter\zen@gobblespace\expandafter\zen@refitem@prop
  \else \expandafter\zen@refitemPrint \expandafter[\expandafter]%
  \fi
}% \zen@refitemProp
```

\zen@refitemPrint is the final macro to print the reference made by \refitem. The properties, if specified, are first analyzed by \zen@refitemText.

```
\long\def\zen@refitemPrint[#1]{\def\zen@@text{#1}%
  \if@tempswa   % got a reference
    \ifx \zen@@text\@empty
        \edef\zen@@text{{\zen@extract\c@zenItem{default}\zen@@noreference}\zen@space}%
    \else\zen@refitemText
    \fi
  \else
    \ifx  \zen@@text\@empty \def\zen@@text{\zen@@noreference\zen@space}%\refused\zen@@label
    \else \zen@refitem@noText
    \fi
  \fi
  \zen@ref@
  \endgroup
}% \zen@refitemPrint
```

\zen@refitemText creates the text to be typeset for the reference, depending on the properties given by the user as the last optional argument to \refitem.

```
\def\zen@refitemText{\zen@propcount
  \ifnum \c@zenidx>\z@   \advance\c@zenidx \m@ne
    \ifnum \c@zenidx>\z@
      \edef\zen@@parents{\zen@extract\c@zenItem{parents}{}}%
      \edef\zen@@depth{\zen@extract\c@zenItem{depth}0}%
    \else \let\zen@@depth \@ne    % shortcut (only one property)
    \fi
    \def\ref{\@ifstar{\zen@ancestorprop{starlabel}}{\zen@ancestorprop{ref}}}%
    \def\label{\@ifstar{\zen@ancestorprop{starlabel}}{\zen@ancestorprop{label}}}%
    \def\default{\zen@ancestorprop{default}}\def\counter{\zen@ancestorprop{counter}}%
    \def\type{\zen@ancestorprop{type}}\def\listname{\zen@ancestorprop{listname}}%
    \def\index{\zen@ancestorprop{index}}\def\listctr{\zen@ancestorprop{listctr}}%
    \def\depth{\zen@ancestorprop{depth}}\def\refitem{\zen@ancestorprop{refitem}}%
    \def\anchor{\zen@ancestorprop{anchor}}\def\commaref{\zen@ancestorprop{commaref}}%
    \def\page{\zen@ancestorprop{page}}\def\currentlabel{\zen@ancestorprop{currentlabel}}%
    \def\currentlabelname{\zen@ancestorprop{currentlabelname}}%
    {\zen@box{\zen@@text
          \expandafter}\expandafter}\expandafter\zen@toks
              \expandafter{\the\zen@toks \zen@nil}%
    \def\ref{\the\zen@toks}\zen@clprops
    \def\ref{\@ifstar{\the\zen@toks}{\the\zen@toks}}\let\label\ref
    \long\def\zen@##1##2\zen@nil{\zen@toks{##2\zen@nil}##1}\let\zen@nil\relax
  \fi
}% \zen@refitemText
\def\zen@ancestorprop#1{%
  \ifnum \zen@@depth>\c@zenidx
    \advance\c@zenidx \m@ne
    \ifnum \c@zenidx<\z@   \let\zen@ \c@zenItem
    \else                  \edef\zen@{\zen@ancestor}%
    \fi
    \edef\zen@temp{{\zen@extract\zen@{#1}\zen@@noreference}}%
  \else \def\zen@temp{{}}\advance\c@zenidx \m@ne
  \fi
  \zen@toks\expandafter{\the\expandafter\zen@toks \expandafter\zen@ \zen@temp}%
}% \zen@ancestorproperty
\def\zen@ancestor{\expandafter\ltx@car\romannumeral0\expandafter\ltx@GobbleNum
  \expandafter{\the\expandafter\c@zenidx\expandafter}\zen@@parents{}{}{}\@nil % no need of {}...
}% \zen@getancestor
\def\zen@propcount{\c@zenidx\z@
  \def\ref{\advance\c@zenidx \@ne}\zen@clprops
  {\zen@box{\zen@@text \expandafter}\expandafter}\expandafter
                        \c@zenidx \the\c@zenidx\relax
}% \zen@propcount
\def\zen@clprops{\let\label\ref   \let\page\ref
  \let\none\ref  \let\default\ref \let\counter\ref \let\listname\ref
```

```
  \let\anchor\ref \let\depth\ref  \let\type\ref    \let\index\ref   \let\listctr\ref
  \let\refitem\ref \let\commaref\ref \let\currentlabel\ref \let\currentlabelname\ref
}% \zen@clprops
\def\zen@refitem@noText{\let\ref \zen@@noreference \zen@clprops}
```

\zen@refitemExternal for references to an anchor in an external document with \hyperref:

```
\def\zen@refitemExternal[#1]#2{%
  \toks@{#1}\zen@Normalize\zen@lowercase\zen@@label {#2}%
  \edef\zen@ref@{\endgroup
    \noexpand\hyperref{\the\toks@}{refitem}{\zen@temp}%
  }\zen@ref@
}% \zen@Hy@external
```

4.11 Undefined references and duplicate labels management

\zen@MultipleLabels

```
\def\zen@MultipleLabels{%
  \PackageWarning{enumitem-zref}
    {‘\zen@@label’ on page \thepage \space multiple defined\MessageBreak
     Only the first one will be referenced}% et mettre ici la liste des références correspondant aux alias
}% \zen@MultipleLabels
```

4.12 \zeninfo (If you are lost...)

\zen@info \zeninfo executes \zen@info when inside environments:

```
\newrobustcmd*\zen@info[1][]{\begingroup \sloppy
  \def\sep##1{##1\hskip\z@ plus.001fil\relax\linebreak[2]}%
  \ifcase\zen@whichinfo{#1}  \zen@@depth      % case 0 (depth)
  \or                    \the\c@zenidx    % case 1 (index)
  \or                                     % case 2 (type)
    \ifcase\zen@@type enumerate\or itemize\or description\or unknown\fi
  \or anchor=\ifzen@Hy\zen@@anchor\else no hyperref\fi % case 3 (anchor)
  \or refitem=\sep{}\detokenize\expandafter{\zen@@refitem}% case 4 (refitem)
  \or commaref=\sep{}\detokenize\expandafter{\zen@@commaref}% case 5 (commaref)
  \else                                   % (case -1) give all infos
    {\ifdefined\scriptsize\scriptsize\fi
       zeninfo:\sep{}depth=\sep{}\zen@@depth
             \sep/ idx=\sep{}\the\c@zenidx
       \ifcase\zen@@type \sep/ ctr=\sep{}\the\value\@listctr \fi % enumerate only
             \sep/ commaref=\sep{}\detokenize\expandafter{\zen@@commaref}%
             \sep/ refitem=\sep{}\detokenize\expandafter{\zen@@refitem}%
       \ifzen@Hy\sep/ anchor=\sep{}\zen@@anchor \fi
    }%
  \fi\endgroup
}% \zen@info
```

\zeninfo is executed when out of scope:

```
\newrobustcmd\zeninfo[1][]{\begingroup
  \ifdefined\zen@@type
  \ifcase \zen@@type \zeninfo@err
Ï  \or             \zeninfo@err
Ð  \or             \zeninfo@err
  \else\zeninfo@err[{in environment \@currenvir\MessageBreak
                     You should be in a list environment}]%
  \fi\endgroup
}% \zeninfo (out of scope)
\newcommand*\zeninfo@err[1][{you should give a name to your \@currenvir\space list}]{%
  \PackageError{enumitem-zref}
    {\string\zeninfo\space is not available\MessageBreak
    #1}\@eha
}% \zeninfo@err
```

\zen@whichinfo

```
\def\zen@whichinfo#1{%
  \expandafter\strip@prefix     \if d#1>\z@   % depth  (begins with  d)
  \else>\expandafter\strip@prefix\if i#1>\@ne  % index  (begins with  i)
  \else>\expandafter\strip@prefix\if t#1>\tw@  % type   (begins with  t)
  \else>\expandafter\strip@prefix\if a#1>\thr@@% anchor (begins with  a)
  \else>\expandafter\strip@prefix\if r#1>4     % refitem(begins with  r)
  \else>\expandafter\strip@prefix\if c#1>5     % commaref(begins with c)
  \else>\m@ne
  \fi\fi\fi\fi\fi\fi
}% \zen@whichinfo
```

4.13 Create the references At Begin Document for the *refitem* and the *commaref* schemes

\zen@item@comma

```
\def\zen@item@comma{%
  \begingroup \c@zenidx \@ne %%\loggingall
  \@whilesw \ifcsname Z@R@zen>\the\c@zenidx\endcsname\fi
                {\advance\c@zenidx \@ne}%
  \@whilenum \c@zenidx>\@ne
  \do{\advance \c@zenidx \m@ne
    \zref@wrapper@unexpanded{%
    \edef\zen@refitem{%
        \zref@extractdefault{zen>\the\c@zenidx}{zen@refitem}{}%
        \zref@extractdefault{zen>\the\c@zenidx}{zen@commaref}{}}%
     \edef\zen@refitem{\expandafter
        \zen@item@comma@ \zen@refitem \@nnil}%
     \expandafter\zen@def@item@comma@ \zen@refitem \@nnil}}%
  \endgroup
}% \zen@item@comma
\def\zen@item@comma@ #1{%
  \ifx\@nnil#1\else
    \expandafter\noexpand\csname zen@refitem(\detokenize{#1})\endcsname
    \expandafter\zen@item@comma@ % (loop)
  \fi
}% \zen@item@comma@
```

\zen@def@item@comma@

```
\def\zen@def@item@comma@ #1{\ifx\@nnil#1\else
    \zen@def@item@comma{#1}\expandafter\zen@def@item@comma@ \fi % loop
}% \zen@def@item@comma@
```

\zen@def@item@comma

```
\newif\ifzen@duplicatealiases
\def\zen@def@item@comma #1{%
  \xdef#1{%
    \ifdefined#1%
      \ifx\relax#1\@empty       \@ne>\the\c@zenidx% create new
      \else \expandafter
        \ifnum\expandafter\strip@prefix#1=\c@zenidx% redefinition to the same location
          #1%
        \else
          \number\numexpr1+#1% increment multiplicity
        \fi
      \fi
    \else                  \@ne>\the\c@zenidx% create new
    \fi}%
  \ifzen@duplicatealiases\else
  \expandafter\expandafter\expandafter\strip@prefix
    \expandafter\strip@prefix\ifnum\@ne<#1>\global\zen@duplicatealiasestrue\else>>\fi
  \fi
```

```
}% \zen@def@item@comma
\zen@AtEnd \let\zen@AtEnd\@undefined
⟨/package⟩
```

References

[1] *The enumitem package*; Javier Bezos
2009/05/18 v2.2 – Customized lists
CTAN:help/Catalogue/entries/enumitem.html

[2] *The zref package*; Heiko Oberdiek
2010/05/01 v2.17 – New reference scheme for LaTeX2e
CTAN:help/Catalogue/entries/zref.html

[3] *The gettitlestring package*; Heiko Oberdiek
2009/12/18 v1.3 – Cleanup title references
CTAN:help/Catalogue/entries/gettitlestring.html

[4] *The engrec package*; Yvon Henel
2008/05/07 v1.1 – Greek letters from counters
CTAN:help/Catalogue/entries/engrec.html

History

[2011/02/18 v1.8]

- Documentation recompiled after tabu[9] package v2.5 release.

[2010/12/30 v1.75]

- Bug fixed with empty items and nested lists.

[2010/12/27 v1.7]

- Documentation improvement.
- No modification in the package file.

[2010/12/17 v1.5]

- enumitem-zref now works with the keys: `start=`, `resume`, and `resume*` of enumitem package.
- Package option: `greekctr` added as an alternative to the `greek` package option: `greekctr` is based on the greekctr[10] package while `greek` is based on engrec[11]. Both uses code of alphalph[12] in order to make the counters *wrap* around the greek alphabet.
  Options `greek` and/or `greekctr` are automatically loaded `\AtBeginDocument` if the engrec / greekctr packages are detected.

[2010/12/10 v1.2]

- Bug encountered in `\zen@anchors@`:
  `\ifx#1\@nnil` is not the same as `\ifx \@nnil#1` !!

[2010/12/02 v1.1]

- The first version.

9. tabu: CTAN:help/Catalogue/entries/tabu.html
10greekctr: CTAN:help/Catalogue/entries/greekctr.html
11engrec: CTAN:help/Catalogue/entries/engrec.html
12alphalph: CTAN:help/Catalogue/entries/alphalph.html

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Ë

Ï

Đ