

# The zref package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2009/12/08 v2.7

## Abstract

Package `zref` tries to get rid of the restriction in  $\text{\LaTeX}$ 's reference system that only two properties are supported. The package implements an extensible referencing system, where properties are handled in a more flexible way. It offers an interface for macro programmers for the access to the system and some applications that uses the new reference scheme.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Standard $\text{\LaTeX}$ behaviour	3
1.2	Basic idea	3
1.3	Interfaces	4
<b>2</b>	<b>Interface for programmers</b>	<b>4</b>
2.1	Entities	4
2.2	Property list	5
2.3	Property	5
2.4	Reference generation	6
2.5	Data extraction	6
2.6	Setup	7
2.7	Declared properties	7
2.8	Wrapper for advanced situations	7
2.9	Counter for unique names	8
<b>3</b>	<b>User interface</b>	<b>8</b>
3.1	Module <code>user</code>	8
3.2	Module <code>abspage</code>	9
3.3	Module <code>lastpage</code>	9
3.3.1	Example	9
3.4	Module <code>totpages</code>	10
3.5	Module <code>runs</code>	10
3.6	Module <code>perpage</code>	11
3.7	Module <code>counter</code>	11
3.8	Module <code>titleref</code>	11
3.9	Module <code>savepos</code>	12
3.10	Module <code>dotfill</code>	12
3.11	Module <code>xr</code>	13
<b>4</b>	<b>ToDo</b>	<b>13</b>
<b>5</b>	<b>Example</b>	<b>14</b>

<b>6</b>	<b>Implementation</b>	<b>16</b>
6.1	Package <code>zref</code>	16
6.1.1	Identification	16
6.1.2	Load basic module	16
6.1.3	Process options	16
6.2	Module <code>base</code>	17
6.2.1	Prefixes	17
6.2.2	Identification	17
6.2.3	Utilities	17
6.2.4	Check for $\varepsilon$ - <code>T<sub>E</sub>X</code>	18
6.2.5	Auxiliary file stuff	18
6.2.6	Property lists	19
6.2.7	Properties	20
6.2.8	Reference generation	22
6.2.9	Reference querying and extracting	23
6.2.10	Compatibility with <code>babel</code>	25
6.2.11	Unique counter support	25
6.2.12	Setup	26
6.3	Module <code>user</code>	26
6.4	Module <code>abspage</code>	27
6.5	Module <code>counter</code>	28
6.6	Module <code>lastpage</code>	28
6.7	Module <code>totpages</code>	28
6.8	Module <code>runs</code>	29
6.9	Module <code>perpage</code>	29
6.10	Module <code>titleref</code>	31
6.10.1	Implementation	31
6.10.2	User interface	33
6.10.3	Patches for section and caption commands	33
6.11	Module <code>xr</code>	35
6.12	Module <code>hyperref</code>	39
6.13	Module <code>savepos</code>	40
6.13.1	Identification	40
6.13.2	Availability	40
6.13.3	Setup	40
6.13.4	User macros	40
6.14	Module <code>dotfill</code>	41
<b>7</b>	<b>Test</b>	<b>42</b>
7.1	<code>\zref@localaddprop</code>	42
7.2	Module <code>runs</code>	43
<b>8</b>	<b>Installation</b>	<b>43</b>
8.1	Download	43
8.2	Bundle installation	43
8.3	Package installation	44
8.4	Refresh file name databases	44
8.5	Some details for the interested	44
<b>9</b>	<b>References</b>	<b>45</b>
<b>10</b>	<b>History</b>	<b>45</b>
	[2006/02/20 v1.0]	45
	[2006/05/03 v1.1]	45
	[2006/05/25 v1.2]	45
	[2006/09/08 v1.3]	46
	[2007/01/23 v1.4]	46
	[2007/02/18 v1.5]	46

[2007/04/06 v1.6]	46
[2007/04/17 v1.7]	46
[2007/04/22 v1.8]	46
[2007/05/02 v1.9]	46
[2007/05/06 v2.0]	46
[2007/05/28 v2.1]	46
[2008/09/21 v2.2]	46
[2008/10/01 v2.3]	46
[2009/08/07 v2.4]	47
[2009/12/06 v2.5]	47
[2009/12/07 v2.6]	47
[2009/12/08 v2.7]	47

<b>11 Index</b>	<b>47</b>
-----------------	-----------

## 1 Introduction

Standard L<sup>A</sup>T<sub>E</sub>X’s reference system with `\label`, `\ref`, and `\pageref` supports two properties, the apperance of the counter that is last incremented by `\refstepcounter` and the page with the `\label` command.

Unhappily L<sup>A</sup>T<sub>E</sub>X does not provide an interface for adding another properties. Packages such as `hyperref`, `nameref`, or `titleref` are forced to use ugly hacks to extend the reference system. These ugly hacks are one of the causes for `hyperref`’s difficulty regarding compatibility with other packages.

### 1.1 Standard L<sup>A</sup>T<sub>E</sub>X behaviour

References are created by the `\label` command:

```
\chapter{Second chapter}
\section{First section on page 7} % section 2.1
\label{myref}
```

Now L<sup>A</sup>T<sub>E</sub>X records the section number 2.1 and the page 7 in the reference. Internally the reference is a list with two entries:

```
\r@myref → {2.1}{7}
```

The length of the list if fixed in the L<sup>A</sup>T<sub>E</sub>X kernel, An interface for adding new properties is missing.

There are several tries to add new properties:

**hyperref** uses a list of five properties instead of the standard list with two entries. This causes many compatibility problems with L<sup>A</sup>T<sub>E</sub>X and other packages.

**titleref** stores its title data into the first entry in the list. L<sup>A</sup>T<sub>E</sub>X is happy because it does only see its list with two entries. The situation becomes more difficult, if more properties are added this way. Then the macros form a nested structure inside the first reference argument for the label. Expandable extractions will then become painful.

### 1.2 Basic idea

Some time ago Morten Høgholm sent me an experimental cross referencing mechanism as “expl3” code. His idea is:

```
\g_xref_mylabel_plist →
  \xref_dance_key{salsa}\xref_name_key{Morten}...
```

The entries have the following format:

`\xref_{\langle your key \rangle}_key{\langle some text \rangle}`

This approach is much more flexible:

- New properties can easily be added, just use a new key.
- The length of the list is not fixed. A reference can use a subset of the keys.
- The order of the entries does not matter.

Unhappily I am not familiar with the experimental code for L<sup>A</sup>T<sub>E</sub>X3 that will need some time before its first release. Thus I have implemented it as L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> package without disturbing the existing L<sup>A</sup>T<sub>E</sub>X reference system.

### 1.3 Interfaces

The package provides a generic *interface for programmers*. Commands of this interface are prefixed by `\zref@`.

Option `user` enables the *user interface*. Here the commands are prefixed by `\z` to avoid name clashes with existing macros.

Then the packages provides some *modules*. They are applications for the reference system and can also be considered as examples how to use the reference system.

The modules can be loaded as packages. The package name is prefixed with `zref-`, for example:

`\RequirePackage{zref-abspage}`

This is the preferred way if the package is loaded from within other packages to avoid option clashes.

As alternative package `zref` can be used and the modules are given as options:

`\usepackage[perpage,user]{zref}`

## 2 Interface for programmers

The user interface is described in the next section [3](#).

### 2.1 Entities

**Reference.** Internally a reference is a list of key value pairs:

`\Z@R@myref → \default{2.1}\page{7}`

The generic format of a entry is:

`\Z@R@⟨refname⟩ → \⟨propname⟩{⟨value⟩}`

`⟨refname⟩` is the name that denoted references (the name used in `\label` and `\ref`). `⟨propname⟩` is the name of the property or key. The property key macro is never executed, it is used in parameter text matching only.

**Property.** Because the name of a property is used in a macro name that must survive the `.aux` file, the name is restricted to letters and ‘@’.

**Property list.** Often references are used for special purposes. Thus it saves memory if just the properties are used in this reference that are necessary for its purpose.

Therefore this package uses the concept of *property lists*. A property list is a set of properties. The set of properties that is used by the default `\label` command is the *main property list*.

## 2.2 Property list

<sup>exp</sup> means that the implementation of the marked macro is expandable.

`\zref@newlist {<listname>}`

Declares a new empty property list.

`\zref@addprop {<listname>} {<propname>}`

Adds the property `<propname>` to the property list `<listname>`. The property and list must exist.

`\zref@localaddprop {<listname>} {<propname>}`

Local variant of `\zref@addprop`.

`\zref@listexists {<listname>} {<then>}`

Executes `<then>` if the property list `<listname>` exists or raise an error otherwise.

`\zref@iflistundefinedexp {<listname>} {<then>} {<else>}`

Executes `<then>` if the list exists or `<else>` otherwise.

`\zref@iflistcontainsprop {<listname>} {<propname>} {<then>} {<else>}`

Executes `<then>` if the property `<propname>` is part of property list `<listname>` or otherwise it runs the `<else>` part.

## 2.3 Property

`\zref@newprop * {<propname>} [<default>] {<value>}`

This command declares and configures a new property with name `<propname>`.

In case of unknown references or the property does not exist in the reference, the `<default>` is used as value. If it is not specified here, a global default is used, see `\zref@setdefault`.

The correct values of some properties are not known immediately but at page shipout time. Prominent example is the page number. These properties are declared with the star form of the command.

`\zref@setcurrent {<propname>} {<value>}`

This sets the current value of the property `<propname>`. It is a generalization of setting L<sup>A</sup>T<sub>E</sub>X's `\currentlabel`.

`\zref@getcurrent {<propname>} {<value>}`

This returns the current value of the property `<propname>`. The value may not be correct, especially if the property is bound to a page (start form of `\zref@newprop`) and the right value is only known at shipout time (e.g. property 'page').

`\zref@propexists {⟨propname⟩} {⟨then⟩}`

Calls  $\langle then \rangle$  if the property  $\langle propname \rangle$  is available or generates an error message otherwise.

`\zref@ifpropundefinedexp {⟨propname⟩} {⟨then⟩} {⟨else⟩}`

Calls  $\langle then \rangle$  or  $\langle else \rangle$  depending on the existence of property  $\langle propname \rangle$ .

## 2.4 Reference generation

`\zref@label {⟨refname⟩}`

This works similar to `\label`. The reference  $\langle refname \rangle$  is created and put into the `.aux` file with the properties of the main property list.

`\zref@labelbylist {⟨refname⟩} {⟨listname⟩}`

Same as `\zref@label` except that the properties are taken from the specified property list  $\langle listname \rangle$ .

`\zref@labelbyprops {⟨refname⟩} {⟨propnameA⟩,⟨propnameB⟩,...}`

Same as `\zref@label` except that these properties are used that are given as comma separated list in the second argument.

`\zref@newlabel {⟨refname⟩} {...}`

This is the macro that is used in the `.aux` file. It is basically the same as `\newlabel` apart from the format of the data in the second argument.

## 2.5 Data extraction

`\zref@extractdefaultexp {⟨refname⟩} {⟨propname⟩} {⟨default⟩}`

This is the basic command that references the value of a property  $\langle propname \rangle$  for the reference  $\langle refname \rangle$ . In case of errors such as undefined reference the  $\langle default \rangle$  is used instead.

`\zref@extractexp {⟨refname⟩} {⟨propname⟩}`

The command is an abbreviation for `\zref@extractdefault`. As default the default of the property is taken, otherwise the global default.

Example for page references:

```

LATEX: \pageref{foobar}
zref:    \zref@extract{foobar}{page}
```

Both `\zref@extract` and `\zref@extractdefault` are expandable. That means, these macros can directly be used in expandable calculations, see the example file. On the other side, babel's shorthands are not supported, there are no warnings in case of undefined references.

If an user interface doesn't need expandable macros then it can use `\zref@refused` and `\zref@wrapper@babel` for its user macros.

`\zref@refused {⟨refname⟩}`

This command is not expandable. It causes the warnings if the reference `⟨refname⟩` is not defined. Use the `\zref@extract` commands inside expandable contexts and mark their use outside by `\zref@refused`, see the example file.

`\zref@ifrefundefinedexp {⟨refname⟩} {⟨then⟩} {⟨else⟩}`

A possibility to check whether a reference exists.

`\zref@ifrefcontainspropexp {⟨refname⟩} {⟨propname⟩} {⟨then⟩} {⟨else⟩}`

Test whether a reference provides a property.

## 2.6 Setup

`\zref@default`

Holds the global default for unknown values.

`\zref@setdefault {⟨value⟩}`

Sets the global default for unknown values. The global default is used, if a property does not specify an own default and the value for a property cannot be extracted. This can happen if the reference is unknown or the reference does not have the property.

`\zref@setmainlist {⟨value⟩}`

Sets the name of the main property list. The package sets and uses `main`.

## 2.7 Declared properties

Module	Property	Property list	Default
	<code>default</code>	<code>main</code>	<code>&lt;empty&gt;</code>
	<code>page</code>	<code>main</code>	<code>&lt;empty&gt;</code>
<code>abspage, totpages</code>	<code>abspage</code>	<code>main</code>	0
<code>perpage</code>	<code>pagevalue</code>	<code>perpage</code>	0
	<code>page</code>	<code>perpage</code>	<code>&lt;empty&gt;</code>
	<code>abspage</code>	<code>perpage</code>	0
<code>counter</code>	<code>counter</code>	<code>main</code>	<code>&lt;empty&gt;</code>
<code>titleref</code>	<code>title</code>	<code>main</code>	<code>&lt;empty&gt;</code>
<code>savepos</code>	<code>posx</code>	<code>savepos</code>	0
	<code>posy</code>	<code>savepos</code>	0
<code>hyperref</code>	<code>anchor</code>	<code>main</code>	<code>&lt;empty&gt;</code>
	<code>url</code>		<code>&lt;empty&gt;</code>
<code>xr</code>	<code>url</code>		<code>&lt;empty&gt;</code>

## 2.8 Wrapper for advanced situations

`\zref@wrapper@babel {...} {⟨name⟩}`

This macro helps to add shorthand support. The second argument is protected, then the code of the first argument is called with the protected name appended.

Examples are in the sources.

`\zref@wrapper@immediate {...}`

There are situations where a label must be written instantly to the `.aux` file, for example after the last page. If the `\zlabel` or `\label` command is put inside this wrapper, immediate writing is enabled. See the implementation for module `lastpage` for an example of its use.

`\zref@wrapper@unexpanded {...}`

Assuming someone wants to extract a value for property `bar` and store the result in a macro `\foo` without traces of the expanding macros and without expanding the value. This (theoretical?) problem can be solved by this wrapper:

```
\edef\foo{%
  \zref@wrapper@unexpanded{%
    \zref@extract{someref}{bar}%
  }%
}
```

The `\edef` forces the expansion of `\zref@extract`, but the extraction of the value is prevented by the wrapper that uses  $\varepsilon$ -TEX' `\unexpanded` for this purpose.

## 2.9 Counter for unique names

Some modules (`titleref` and `dotfillmin`) need unique names for automatically generated label names.

`\zref@require@unique`

This command creates the unique counter `zref@unique` if the counter does not already exist.

`\thezref@unique`

This command is used to generate unique label names.

## 3 User interface

### 3.1 Module user

The user interface for this package and its modules is enabled by `zref's` package option `user` or package `zref-user`. The names of user commands are prefixed by `z` in order to avoid name clashes with existing macros of the same functionality. Thus the package does not disturb the traditional reference scheme, both can be used together.

The syntax descriptions contain the following markers that are intended as hints for programmers:

- |        |  |
|--------|--|
| babel  | Babel shorthands are allowed.  |
| robust | Robust macro.  |
| exp    | Expandable version:  |
|        | <ul style="list-style-type: none"> <li>• robust, unless the extracted values are fragile,</li> <li>• no babel shorthand suport.</li> </ul> |



The basic user interface of the package without modules are commands that mimic the standard L<sup>A</sup>T<sub>E</sub>X behaviour of `\label`, `\ref`, and `\pageref`:

`\zlabel {⟨refname⟩}{}babel`

Similar to `\label`. It generates a label with name `⟨refname⟩` in the new reference scheme.

`\zref [⟨propname⟩] {⟨refname⟩}{}babel`

Without optional argument similar to `\ref`, it returns the default reference property. This property is named `default`:

$$\backslash\mathrm{zref}\{x\} \equiv \backslash\mathrm{zref}[\mathrm{default}]\{x\}$$

`\zpageref {⟨refname⟩}{}babel`

Convenience macro, similar to `\pageref`.

$$\backslash\mathrm{zpageref}\{x\} \equiv \backslash\mathrm{zref}[\mathrm{page}]\{x\}$$

`\zrefused {⟨refname⟩}{}babel`

Some of the user commands in the modules are expandable. The use of such commands do not cause any undefined reference warnings, because inside of expandable contexts this is not possible. However, if there is a place outside of expandable contexts, `\refused` is strongly recommended. The reference `⟨refname⟩` is marked as used, undefined ones will generate warnings.

## 3.2 Module `abspage`

With the help of package `atbegshi` a new counter `abspage` with absolute page numbers is provided. Also a new property `abspage` is defined and added to the main property list. Thus you can reference the absolute page number:

```
Section \zref{foo} is on page \zpageref{foo}.
This is page \zref[abspage]{foo} of \zref[abspage]{LastPage}.
```

The example also makes use of module `lastpage`.

## 3.3 Module `lastpage`

Provides the functionality of package `lastpage` [3] in the new reference scheme. The label `LastPage` is put at the end of the document. You can refer the last page number with:

```
\zpageref{LastPage}
```

Since version 2008/10/01 v2.3 the module defines the list `LastPage`. In addition to the properties of the main list label `LastPage` also stores the properties of this list `LastPage`. The default of this list is empty. The list can be used by the user to add additional properties for label `LastPage`.

### 3.3.1 Example

```
1 ⟨*example-lastpage⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \documentclass{report}
4 %
```

```

5 \newcounter{foo}
6 \renewcommand*{\thefoo}{\Alph{foo}}
7 %
8 \usepackage{zref-lastpage,zref-user}[2008/10/01]
9 %
10 \makeatletter
11 \zref@newprop{thefoo}{\thefoo}
12 \zref@newprop{valuefoo}{\the\value{foo}}
13 \zref@newprop{chapter}{\thechapter}
14 \zref@addprop{LastPage}{thefoo}
15 \zref@addprop{LastPage}{valuefoo}
16 \zref@addprop{LastPage}{chapter}
17 \makeatother
18 %
19 \newcommand*{\foo}{%
20   \stepcounter{foo}%
21   [Current foo: \thefoo]%
22 }
23 %
24 \begin{document}
25   \chapter{First chapter}%
26   Last page is \zref{LastPage}.\%
27   Last chapter is \zref[chapter]{LastPage}.\%
28   Last foo is \zref[thefoo]{LastPage}.\%
29   Last value of foo is \zref[valuefoo]{LastPage}.\%
30   \foo
31   \chapter{Second chapter}%
32   \foo\foo\foo
33   \chapter{Last chapter}%
34   \foo
35 \end{document}
36 \</example-lastpage>

```

### 3.4 Module **totpages**

For the total number of pages of a document you need to know the absolute page number of the last page. Both modules **abspage** and **lastpage** are necessary and automatically enabled.

`\ztotpagesexp`

Prints the total number of pages or 0 if this number is not yet known. It expands to an explicit number and can also be used even in expandable calculations (`\numexpr`) or counter assignments.

### 3.5 Module **runs**

Module **runs** counts the  $\text{\LaTeX}$  runs since last `.aux` file creation and prints the number in the `.log` file.

`\zrunsexp`

Prints the total number of  $\text{\LaTeX}$  runs including the current one. It expands to an explicit number. Before `\begin{document}` the value is zero meaning the `.aux` file is not read yet. If a previous `.aux` file exists, the value found there increased by one is the new number. Otherwise `\zruns` is set to one.  $\text{\LaTeX}$  runs where the `.aux` files are not rewritten are not counted (see `\nofiles`).

### 3.6 Module `perpage`

With `\@addtoreset` or `\numberwithin` a counter can be reset if another counter is incremented. This do not work well if the other counter is the page counter. The page counter is incremented in the output routine that is often called asynchronous somewhere on the next page. A reference mechanism costs at least two L<sup>A</sup>T<sub>E</sub>X runs, but ensures correct page counter values.

`\zmakeperpage [<reset>] {<counter>}`

At the of a new page counter *<counter>* starts counting with value *<reset>* (default is 1). The macro has the same syntax and semantics as `\MakePerPage` of package `perpage` [5]. Also `perpage` of package `footmisc` [1] can easily be simulated by

```
\zmakeperpage{footnote} % \usepackage[perpage]{footmisc}
```

If footnote symbols are used, some people dislike the first symbol †. It can easily be skipped:

```
\zmakeperpage[2]{footnote}
```

`\thezpage  
counter zpage`

If the formatted counter value of the counter that is reset at a new page contains the page value, then you can use `\thezpage`, the page number of the current page. Or counter `zpage` can be used, if the page number should be formatted differently from the current page number. Example:

```
\newcounter{foobar}  
\zmakeperpage{foobar}  
\renewcommand*{\thefoobar}{\thezpage-\arabic{foobar}}  
% or  
\renewcommand*{\thefoobar}{\roman{zpage}-\arabic{foobar}}
```

`\zunmakeperpage {<counter>}`

The reset mechanism for this counter is deactivated.

### 3.7 Module `counter`

This option just add the property `counter` to the main property list. The property stores the counter name, that was responsible for the reference. This is the property `hyperref`'s `\autoref` feature uses. Thus this property `counter` may be useful for a reimplementation of the `autoref` feature, see the section 4 with the `todo` list.

### 3.8 Module `titleref`

This option makes section and caption titles available to the reference system similar to packages `titleref` or `nameref`.

`\ztitleref {<refname>}babel`

Print the section or caption title of reference *<refname>*, similar to `\nameref` or `\titleref`.

`\ztitlerefsetup {key1=value1, key2=value2, ...}`

This command allows to configure the behaviour of module `titleref`. The following keys are available:

`title=<value>`

Sets the current title.

`stripperperiod=true|false`

Follow package `nameref` that removes a last period. Default: `true`.

`expand=true|false`

Package `\titleref` expands the title first. This way garbage and dangerous commands can be removed, e.g. `\label`, `\index...`. See implementation section for more details. Default is `false`.

`cleanup={...}`

Hook to add own cleanup code, if method `expand` is used. See implementation section for more details.

### 3.9 Module `savepos`

This option supports a feature that pdfTeX provides (and XeTeX). pdfTeX is able to tell the current position on the page. The page position is not instantly known. First the page must be constructed by TeX's asynchronous output routine. Thus the time where the position is known is the page shipout time. Thus a reference system where the information is recorded in the first run and made available for use in the second run comes in handy.

`\zsavepos {<refname>}`

It generates a reference with name `<refname>` to the location where the command is executed.

`\zposxexp {<refname>}`  
`\zposyexp {<refname>}`

Get the position as number. Unit is sp. Horizontal positions by `\zposx` increase from left to right. Vertical positions by `\zposy` from bottom to top.

Do not rely on absolute page numbers. Because of problems with the origin the numbers may differ in DVI or PDF mode of pdfTeX. Therefore work with relative values by comparisons.

Both `\zposx` and `\zposy` are expandable and can be used inside calculations (`\setcounter`, `\addtocounter`, package `calc`, `\numexpr`). However this property prevents from notifying L<sup>A</sup>T<sub>E</sub>X that the reference is actually used (the notifying is not expandable). Therefore you should mark the reference as used by `\zrefused`.

This module uses pdfTeX's `\pdfsavepos`, `\pdflastxpos`, and `\pdflastypos`. They are available in PDF mode and since version 1.40.0 also in DVI mode.

### 3.10 Module `dotfill`

`\zdotfill`

This package provides the command `\zdotfill` that works similar to `\dotfill`, but can be configured. Especially it suppresses the dots if a minimum number of dots cannot be set.

`\zdotfillsetup {key1=value1, key2=value2, ...}`

This command allows to configure the behaviour of `\zdotfill`. The following keys are available:

`min`= $\langle$ *count value* $\rangle$

If the actual number of dots are smaller than  $\langle$ *count value* $\rangle$ , then the dots are suppressed. Default: 2.

`unit`= $\langle$ *dimen value* $\rangle$

The width of a dot unit is given by  $\langle$ *dimen value* $\rangle$ . Default: 0.44em (same as the unit in `\dotfill`).

`dot`= $\langle$ *value* $\rangle$

The dot itself is given by  $\langle$ *value* $\rangle$ . Default: . (dot, same as the dot in `\dotfill`).

### 3.11 Module `xr`

This package provides the functionality of package `xr`, see [8]. It also supports the syntax of `xr-hyper`.

`\zexternaldocument * [ $\langle$ prefix $\rangle$ ] babel { $\langle$ external document $\rangle$ } [ $\langle$ url $\rangle$ ]`

See `\externaldocument` for a description of this option. The standard reference scheme and the scheme of this package use different name spaces for reference names. If the external document uses both systems. Then one import statement would put the names in one namespace and probably causing problems with multiple references of the same name. Thus the star form only looks for `\newlabel` in the `.aux` files, whereas without star only `\zref@newlabels` are used.

In the star form it tries to detect labels from `hyperref`, `titleref`, and `ntheorem`. If such an extended property from the packages before cannot be found or are empty, they are not included in the imported reference.

Warnings are given if a reference name is already in use and the item is ignored. Unknown properties will automatically be declared.

If the external references contain `anchor` properties, then we need also a url to be able to address the external file. As default the filename is taken with a default extension.

`\zxrsetup {key1=value1, key2=value2, ...}`

Currently the key `ext` is defined, this sets the url default extension.

`\zref@xr@ext`

If the  $\langle$ *url* $\rangle$  is not specified in `\zref@externaldocument`, then the url will be constructed with the file name and this macro as extension. `\XR@ext` is used if `hyperref` is loaded, otherwise `pdf`.

## 4 ToDo

Among other things the following issues are left for future work:

- The user land macros are not checked for robustness yet. They can be fragile. If this happens, use `\protect` until a later version of this package. The `\protect` will not disturb, if the protected macro become robust in the future.
- Other applications: `autoref`, `hyperref`, ...

## 5 Example

```
37 <*example>
38 \documentclass{book}
39
40 \usepackage[ngerman]{babel}%
41
42 \usepackage[savepos,totpages,titleref,dotfill,counter,user]{zref}
43
```

Chapters are wrapped inside `\ChapterStart` and `\ChapterStop`. The first argument #1 of `\ChapterStart` is used to form a label id `chap:#1`. At the end of the chapter another label is set by `\zref@wrapper@immediate`, because otherwise at the end of document a deferred write would not be written, because there is no page for shipout.

Also this example shows how chapter titles can be recorded. A new property `chaptitle` is declared and added to the main property list. In `\ChapterStart` the current value of the property is updated.

```
44 \makeatletter
45 \zref@newprop{chaptitle}{}
46 \zref@addprop{main}{chaptitle}
47
48 \newcommand*{\ChapterStart}[2]{%
49   \cleardoublepage
50   \def\current@chapid{#1}%
51   \zref@setcurrent{chaptitle}{#2}%
52   \chapter{#2}%
53   \zlabel{chap:#1}%
54 }
55 \newcommand*{\ChapterStop}{%
56   \cleardoublepage
57   \zref@wrapper@immediate{%
58     \zref@labelbyprops{chapend:\current@chapid}{abspage}%
59   }%
60 }
```

`\ChapterPages` calculates and returns the number of pages of the referenced chapter.

```
61 \newcommand*{\ChapterPages}[1]{%
62   \zrefused{chap:#1}%
63   \zrefused{chapend:#1}%
64   \number\numexpr
65     \zref@extract{chapend:#1}{abspage}%
66     -\zref@extract{chap:#1}{abspage}%
67   +1\relax
68 }
69 \makeatother
70 \begin{document}
```

As exception we use `\makeatletter` here, because this is just an example file that also should show some of programmer's interface.

```
71 \makeatletter
72
73 \frontmatter
74 \zlabel{documentstart}
75
76 \begin{itemize}
77 \item
78   The frontmatter part has
79   \number\numexpr\zref@extract{chap:first}{abspage}-1\relax~pages.
80 \item
81   Chapter \zref{chap:first} has \ChapterPages{first} page(s).
82 \item
```

```

83 Section \zref{hello} is on the
84 \ifcase\numexpr
85   \zref@extractdefault{hello}{page}{0}%
86   -\zref@extractdefault{chap:first}{page}{0}%
87   +1\relax
88   ??\or first\or second\or third\or forth\fi
89   ~page inside its chapter.
90 \item
91   The document has
92   \zref[abspage]{LastPage} pages.
93   This number is \ifodd\ztotpages odd\else even\fi.
94 \item
95   The last page is labeled with \zpageref{LastPage}.
96 \item
97   The title of chapter \zref{chap:next} is ‘‘\zref[chaptitle]{chap:next}’’.
98 \end{itemize}
99
100 \tableofcontents
101
102 \mainmatter
103 \ChapterStart{first}{First chapter}
104

```

The user level commands should protect babel shorthands where possible. On the other side, expandable extracting macros are useful in calculations, see above the examples with `\numexpr`.

```

105 \section{Test}
106 \zlabel{a"o}
107 Section \zref{a"o} on page
108 \zref@wrapper@babel\zref@extract{a"o}{page}.
109
110 Text.
111 \newpage
112
113 \section{Hello World}
114 \zlabel{hello}
115
116 \ChapterStop
117
118 \ChapterStart{next}{Next chapter with \emph{umlauts}: "a"o"u"s}
119

```

Here an example follows that makes use of pdfTeX’s “savepos” feature. The position on the page is not known before the page is constructed and shipped out. Therefore the position is stored in references and are available for calculations in the next L<sup>A</sup>T<sub>E</sub>X compile run.

```

120 The width of the first column is
121   \the\dimexpr \zposx{secondcol}sp - \zposx{firstcol}sp\relax,\!
122 the height difference of the two baselines is
123   \the\dimexpr \zposy{firstcol}sp - \zposy{secondline}sp\relax:\!
124 \begin{tabular}{ll}
125   \zsavepos{firstcol}Hello&\zsavepos{secondcol}World\!
126   \zsavepos{secondline}Second line&foobar\!
127 \end{tabular}
128

```

With `\zrefused` L<sup>A</sup>T<sub>E</sub>X is notified, if the references are not yet available and L<sup>A</sup>T<sub>E</sub>X can generate the rerun hint.

```

129 \zrefused{firstcol}
130 \zrefused{secondcol}
131 \zrefused{secondline}
132
133 \ChapterStop

```

Test for module `\dotfill`.

```

134 \ChapterStart{dotfill}{Test for dotfill feature}
135 \newcommand*{\dfptest}[1]{%
136   #1&
137   [\makebox[#1]{\dotfill}]&
138   [\makebox[#1]{\zdotfill}]\
139 }
140 \begin{tabular}{rll}
141 & [\verb|\dotfill|] & [\verb|\zdotfill|]\
142 \dfptest{0.43em}
143 \dfptest{0.44em}
144 \dfptest{0.45em}
145 \dfptest{0.87em}
146 \dfptest{0.88em}
147 \dfptest{0.89em}
148 \dfptest{1.31em}
149 \dfptest{1.32em}
150 \dfptest{1.33em}
151 \end{tabular}
152 \ChapterStop
153 \end{document}
154 \example

```

## 6 Implementation

### 6.1 Package zref

#### 6.1.1 Identification

```

155 \package
156 \NeedsTeXFormat{LaTeX2e}
157 \ProvidesPackage{zref}
158 [2009/12/08 v2.7 New reference scheme for LaTeX2e (HO)]%

```

#### 6.1.2 Load basic module

```

159 \RequirePackage{zref-base}[2009/12/08]

```

Abort package loading if zref-base could not be loaded successfully.

```

160 \@ifundefined{ZREF@baseok}{\endinput}{}

```

#### 6.1.3 Process options

Known modules are loaded and the release date is checked.

```

161 \def\ZREF@temp#1{%
162   \DeclareOption{#1}{%
163     \AtEndOfPackage{%
164       \RequirePackage{zref-#1}[2009/12/08]%
165     }%
166   }%
167 }
168 \ZREF@temp{abspage}
169 \ZREF@temp{counter}
170 \ZREF@temp{dotfill}
171 \ZREF@temp{hyperref}
172 \ZREF@temp{lastpage}
173 \ZREF@temp{perpage}
174 \ZREF@temp{savepos}
175 \ZREF@temp{titleref}
176 \ZREF@temp{totpages}
177 \ZREF@temp{user}
178 \ZREF@temp{xr}
179 \ProcessOptions\relax
180 \package

```



## 6.2 Module base

### 6.2.1 Prefixes

This package uses the following prefixes for macro names:

`\zref@`: Macros of the programmer's interface.

`\ZREF@`: Internal macros.

`\Z@L@listname`: The properties of the list  $\langle listname \rangle$ .

`\Z@D@propname`: The default value for property  $\langle propname \rangle$ .

`\Z@E@propname`: Extract function for property  $\langle propname \rangle$ .

`\Z@X@propname`: Information whether a property value for property  $\langle propname \rangle$  is expanded immediately or at shipout time.

`\Z@C@propname`: Current value of the property  $\langle propname \rangle$ .

`\Z@R@labelname`: Data for reference  $\langle labelname \rangle$ .

`\ZREF@org@`: Original versions of patched commands.

`\z`: For macros in user land, defined if module `user` is set.

The following family names are used for keys defined according to the `keyval` package:

`ZREF@TR`: Setup for module `titleref`.

### 6.2.2 Identification

```
181 <*base>
182 \NeedsTeXFormat{LaTeX2e}
183 \ProvidesPackage{zref-base}%
184 [2009/12/08 v2.7 Module base for zref (HO)]%
```

### 6.2.3 Utilities

`\ZREF@name` Several times the package name is used, thus we store it in `\ZREF@name`.

```
185 \def\ZREF@name{zref}
```

`\ZREF@ErrorNoLine` An error message for this package without line information is generated by `\ZREF@ErrorNoLine`

```
186 \def\ZREF@ErrorNoLine#1#2{%
187   \begingroup
188     \let\on@line\@empty
189     \PackageError\ZREF@name{#1}{#2}%
190   \endgroup
191 }
```

`\ZREF@UpdatePdfTeX` `\ZREF@UpdatePdfTeX` is used as help message text in error messages.

```
192 \def\ZREF@UpdatePdfTeX{Update pdfTeX.}
```

`\ifZREF@found` The following switch is used in list processing.

```
193 \newif\ifZREF@found
```

`\ZREF@patch` Macro `\ZREF@patch` first checks the existence of the command and safes it.

```
194 \def\ZREF@patch#1{%
195   \begingroup\expandafter\expandafter\expandafter\endgroup
196   \expandafter\ifx\csname #1\endcsname\relax
197     \expandafter\@gobble
198   \else
```

```

199 \expandafter\let\csname ZREF@org@#1\expandafter\endcsname
200 \csname #1\endcsname
201 \expandafter\@firstofone
202 \fi
203 }

```

#### 6.2.4 Check for $\varepsilon$ -TeX

The use of  $\varepsilon$ -TeX should be standard nowadays for L<sup>A</sup>T<sub>E</sub>X. We test for  $\varepsilon$ -TeX in order to use its features later.

```

204 \begingroup
205 \ifundefined{eTeXversion}{%
206   \ZREF@ErrorNoLine{%
207     Missing support for eTeX; package is abandoned%
208   }{%
209     Use a TeX compiler that support eTeX and enable eTeX %
210     in the format.%
211   }%
212 \endgroup
213 \endinput
214 }{%
215 \endgroup

216 \RequirePackage{etexcmds}[2007/09/09]
217 \ifetex@unexpanded
218 \else
219   \ZREF@ErrorNoLine{%
220     Missing e-TeX's \string\unexpanded.\MessageBreak
221     Add \string\RequirePackage\string{etexcmds\string} before %
222     \string\documentclass%
223   }{%
224     Probably you are using some package (e.g. ConTeXt) that %
225     redefines \string\unexpanded%
226   }%
227 \expandafter\endinput
228 \fi

```

#### 6.2.5 Auxiliary file stuff

We are using some commands in the .aux files. However sometimes these auxiliary files are interpreted by L<sup>A</sup>T<sub>E</sub>X processes that haven't loaded this package (e.g. package xr). Therefore we provide dummy definitions.

```

229 \RequirePackage{auxhook}
230 \AddLineBeginAux{%
231   \string\providecommand\string\zref@newlabel[2]{}%
232 }

```

`\zref@newlabel` For the implementation of `\zref@newlabel` we call the same internal macro `\@newl@bel` that is used in `\newlabel`. Thus we have for free:

- `\Z@R@labelname` is defined.
- L<sup>A</sup>T<sub>E</sub>X's check for multiple references.
- L<sup>A</sup>T<sub>E</sub>X's check for changed references.

```

233 \def\zref@newlabel{%
234   \@newl@bel{\Z@R}%
235 }

```

## 6.2.6 Property lists

`\zref@newlist` Property lists are stored as list of property names enclosed in curly braces. `\zref@newlist` creates a new list as empty list. Assignments to property lists are global.

```

236 \def\zref@newlist#1{%
237   \zref@iflistundefined{#1}{%
238     \ifdefinable{Z@L@#1}{%
239       \global\expandafter\let\csname Z@L@#1\endcsname\@empty
240       \PackageInfo{zref}{New property list: #1}%
241     }%
242   }{%
243     \PackageError{ZREF@name}{%
244       Property list ‘#1’ already exists%
245     }\@ehc
246   }%
247 }
```

`\zref@iflistundefined` `\zref@iflistundefined` checks the existence of the property list #1. If the property list is present, then #2 is executed and #3 otherwise.

```

248 \def\zref@iflistundefined#1{%
249   \expandafter\ifx\csname Z@L@#1\endcsname\relax
250   \expandafter\@firstoftwo
251   \else
252   \expandafter\@secondoftwo
253   \fi
254 }
```

`\zref@listexists` `\zref@listexists` only executes #2 if the property list #1 exists and raises an error message otherwise.

```

255 \def\zref@listexists#1{%
256   \zref@iflistundefined{#1}{%
257     \PackageError{ZREF@name}{%
258       Property list ‘#1’ does not exist%
259     }\@ehc
260   }%
261 }
```

`\zref@iflistcontainsprop` `\zref@iflistcontainsprop` checks, whether a property #2 is already present in a property list #1.

```

262 \def\zref@iflistcontainsprop#1{%
263   \expandafter\ZREF@iflistcontainsprop\csname Z@L@#1\endcsname
264 }
265 \def\ZREF@iflistcontainsprop#1#2{%
266   \begingroup
267   \ZREF@foundfalse
268   \edef\y{#2}%
269   \expandafter\@tfor\expandafter\x
270   \expandafter:\expandafter=#1\do{%
271     \edef\x{\x}%
272     \ifx\x\y
273       \ZREF@foundtrue
274     \fi
275   }%
276   \expandafter\endgroup
277   \ifZREF@found
278   \expandafter\@firstoftwo
279   \else
280   \expandafter\@secondoftwo
281   \fi
282 }
```

`\zref@listforloop`

```

283 \def\zref@listforloop#1#2{%
284   \expandafter\expandafter\expandafter\@tfor
285   \expandafter\expandafter\expandafter\zref@prop
286   \expandafter\expandafter\expandafter:%
287   \expandafter\expandafter\expandafter=%
288   \csname Z@L@#1\endcsname
289   \do{%
290     #2\zref@prop
291   }%
292 }

```

`\zref@addprop` `\zref@addprop` adds the property #2 to the property list #1, if the property is not already in the list. Otherwise a warning is given.

```

293 \def\zref@addprop#1#2{%
294   \zref@listexists{#1}{%
295     \zref@propexists{#2}{%
296       \zref@iflistcontainsprop{#1}{#2}{%
297         \PackageWarning\ZREF@name{%
298           Property ‘#2’ is already in list ‘#1’%
299         }%
300       }{%
301         \expandafter\g@addto@macro\csname Z@L@#1\endcsname{{#2}}%
302       }%
303     }%
304   }%
305 }

```

`\zref@localaddprop`

```

306 \def\zref@localaddprop#1#2{%
307   \zref@listexists{#1}{%
308     \zref@propexists{#2}{%
309       \zref@iflistcontainsprop{#1}{#2}{%
310         \PackageWarning\ZREF@name{%
311           Property ‘#2’ is already in list ‘#1’%
312         }%
313       }{%
314         \expandafter\ZREF@l@addto@macro\csname Z@L@#1\endcsname{{#2}}%
315       }%
316     }%
317   }%
318 }

```

`\ZREF@l@addto@macro`

```

319 \ifetex@unexpanded
320   \def\ZREF@l@addto@macro#1#2{%
321     \global\let\ZREF@gtemp#1%
322     \g@addto@macro\ZREF@gtemp{#2}%
323     \let#1\ZREF@gtemp
324   }%
325 \else
326   \def\ZREF@l@addto@macro#1#2{%
327     \edef#1{%
328       \etex@unexpanded\expandafter{#1#2}%
329     }%
330   }%
331 \fi

```

## 6.2.7 Properties

`\zref@ifpropundefined` `\zref@ifpropundefined` checks the existence of the property #1. If the property is present, then #2 is executed and #3 otherwise.

```

332 \def\zref@ifpropundefined#1{%
333   \expandafter\ifx\csname Z@E@#1\endcsname\relax
334     \expandafter\@firstoftwo
335   \else
336     \expandafter\@secondoftwo
337   \fi
338 }

```

**\zref@propexists** Some macros rely on the existence of a property. **\zref@propexists** only executes #2 if the property #1 exists and raises an error message otherwise.

```

339 \def\zref@propexists#1{%
340   \zref@ifpropundefined{#1}{%
341     \PackageError\ZREF@name{%
342       Property ‘#1’ does not exist%
343     }\@ehc
344   }%
345 }

```

**\zref@newprop** A new property is declared by **\zref@newprop**, the property name *<propname>* is given in #1. The property is created and configured. If the star form is given, then the expansion of the property value is delayed to page shipout time, when the reference is written to the .aux file.

**\Z@D@propname**: Stores the default value for this property.

**\Z@E@propname**: Extract function.

**\Z@X@propname**: Information whether the expansion of the property value is delayed to shipout time.

**\Z@C@propname**: Current value of the property.

```

346 \def\zref@newprop{%
347   \ifstar{%
348     \let\ZREF@X\noexpand
349     \ZREF@newprop
350   }{%
351     \let\ZREF@X\@empty
352     \ZREF@newprop
353   }%
354 }
355 \def\ZREF@newprop#1{%
356   \PackageInfo{zref}{New property: #1}%
357   \def\ZREF@P{#1}%
358   \ifnextchar[\ZREF@@newprop{\ZREF@@newprop[\zref@default]}%
359 }
360 \def\ZREF@@newprop[#1]{%
361   \global\@namedef{Z@D@\ZREF@P}{#1}%
362   \global\expandafter\let\csname Z@X@\ZREF@P\endcsname\ZREF@X
363   \expandafter\ZREF@@@newprop\csname\ZREF@P\endcsname
364   \zref@setcurrent\ZREF@P
365 }
366 \def\ZREF@@@newprop#1{%
367   \expandafter\gdef\csname Z@E@\ZREF@P\endcsname##1##2##3\ZREF@nil{##2}%
368 }

```

**\zref@setcurrent** **\zref@setcurrent** sets the current value for a property.

```

369 \def\zref@setcurrent#1{%
370   \expandafter\def\csname Z@C@#1\endcsname
371 }

```

**\zref@getcurrent** **\zref@getcurrent** gets the current value for a property.

```

372 \def\zref@getcurrent#1{%
373   \csname Z@C@#1\endcsname
374 }

```

## 6.2.8 Reference generation

`\zref@label` Label macro that uses the main property list.

```
375 \def\zref@label#1{%
376   \zref@labelbylist{#1}\ZREF@mainlist
377 }
```

`\zref@labelbylist` Label macro that stores the properties, specified in the property list #2.

```
378 \def\zref@labelbylist#1#2{%
379   \@bsphack
380   \zref@listexists{#2}{%
381     \expandafter\expandafter\expandafter\ZREF@label
382     \expandafter\expandafter\expandafter{%
383       \csname Z@L@#2\endcsname
384     }{#1}%
385   }%
386   \@esphack
387 }
```

`\zref@labelbyprops` The properties are directly specified in a comma separated list.

```
388 \def\zref@labelbyprops#1#2{%
389   \@bsphack
390   \begingroup
391     \edef\l{#2}%
392     \toks@{}%
393     \@for\x:=#2\do{%
394       \zref@ifpropundefined{\x}{%
395         \PackageWarning\ZREF@name{%
396           Property ‘\x’ is not known%
397         }%
398       }{%
399         \toks@\expandafter\expandafter\expandafter{%
400           \expandafter\the\expandafter\toks@\expandafter{\x}%
401         }%
402       }%
403     }%
404     \expandafter\endgroup
405     \expandafter\ZREF@label\expandafter{\the\toks@}{#1}%
406   \@esphack
407 }
```

`\ifZREF@immediate` The switch `\ifZREF@immediate` tells us, whether the label should be written immediately or at page shipout time. `\ZREF@label` need to be notified about this, because it must disable the deferred execution of property values, if the label is written immediately.

```
408 \newif\ifZREF@immediate
```

`\zref@wrapper@immediate` The argument of `\zref@wrapper@immediate` is executed inside a group where `\write` is redefined by adding `\immediate` before its execution. Also `\ZREF@label` is notified via the switch `\ifZREF@immediate`.

```
409 \long\def\zref@wrapper@immediate#1{%
410   \begingroup
411     \ZREF@immediatetrue
412     \let\ZREF@org@write\write
413     \def\write{\immediate\ZREF@org@write}%
414     #1%
415   \endgroup
416 }
```

`\ZREF@label` `\ZREF@label` writes the data in the `.aux` file. #1 contains the list of valid properties, #2 the name of the reference. In case of immediate writing, the deferred execution of property values is disabled. Also `\ZREF@label` is made expandable in this case.

```

417 \def\ZREF@label#1#2{%
418   \if@filesw
419     \begingroup
420     \ifZREF@immediate
421       \let\ZREF@org@thepage\thepage
422     \fi
423     \protected@write\@auxout{%
424       \ifZREF@immediate
425         \let\thepage\ZREF@org@thepage
426       \fi
427       \let\ZREF@temp\@empty
428       \@tfor\ZREF@P:=#1\do{%
429         \expandafter\ifx
430           \csname\ifZREF@immediate relax\else Z@X@\ZREF@P\fi\endcsname
431           \noexpand
432         \expandafter\let\csname Z@C@\ZREF@P\endcsname\relax
433       \fi
434       \toks@\expandafter{\ZREF@temp}%
435       \edef\ZREF@temp{%
436         \the\toks@
437         \expandafter\string\csname\ZREF@P\endcsname{%
438           \expandafter\noexpand\csname Z@C@\ZREF@P\endcsname
439         }%
440       }%
441     }%
442   }{%
443     \string\zref@newlabel{#2}{\ZREF@temp}%
444   }%
445   \endgroup
446 \fi
447 }
448 \def\ZREF@addtoks#1{%
449   \toks@\expandafter\expandafter\expandafter{%
450     \expandafter\the\expandafter\toks@#1%
451   }%
452 }

```

## 6.2.9 Reference querying and extracting

Design goal for the extracting macros is that the extraction process is full expandable. Thus these macros can be used in expandable contexts. But there are problems that cannot be solved by full expandable macros:

- In standard L<sup>A</sup>T<sub>E</sub>X undefined references sets a flag and generate a warning. Both actions are not expandable.
- Babel's support for its shorthand uses commands that use non-expandable assignments. However currently there is hope, that primitives are added to pdfT<sub>E</sub>X that allows the detection of contexts. Then the shorthand can detect, if they are executed inside `\csname` and protect themselves automatically.

`\zref@ifrefundefined` If a reference #1 is undefined, then macro `\zref@ifrefundefined` calls #2 and #3 otherwise.

```

453 \def\zref@ifrefundefined#1{%
454   \expandafter\ifx\csname Z@R@#1\endcsname\relax
455     \expandafter\@firstoftwo
456   \else
457     \expandafter\@secondoftwo
458   \fi
459 }

```

`\zref@refused` The problem with undefined references is addressed by the macro `\zref@refused`. This can be used outside the expandable context. In case of an undefined reference the flag is set to notify  $\text{\LaTeX}$  and a warning is given.

```
460 \def\zref@refused#1{%
461   \zref@wrapper@babel\ZREF@refused{#1}%
462 }
```

`\ZREF@refused`

```
463 \def\ZREF@refused#1{%
464   \zref@ifrefundefined{#1}{%
465     \protect\G@refundefinedtrue
466     \@latex@warning{%
467       Reference ‘#1’ on page \thepage \space undefined%
468     }%
469   }{%}%
470 }
```

`\zref@extract` `\zref@extract` is an abbreviation for the case that the default of the property is used as default value.

```
471 \def\zref@extract#1#2{%
472   \expandafter\expandafter\expandafter\ZREF@extract
473   \expandafter\expandafter\expandafter{%
474     \csname Z@D@#2\endcsname
475   }{#1}{#2}%
476 }
477 \def\ZREF@extract#1#2#3{%
478   \zref@extractdefault{#2}{#3}{#1}%
479 }
```

`\zref@ifrefcontainsprop` `\zref@ifrefcontainsprop` looks, if the reference #1 has the property #2 and calls then #3 and #4 otherwise.

```
480 \def\zref@ifrefcontainsprop#1#2{%
481   \zref@ifrefundefined{#1}{%
482     \@secondoftwo
483   }{%
484     \expandafter\ZREF@ifrefcontainsprop
485     \csname Z@E@#2\expandafter\endcsname
486     \csname#2\expandafter\expandafter\expandafter\endcsname
487     \expandafter\expandafter\expandafter{%
488       \csname Z@R@#1\endcsname
489     }%
490   }%
491 }
492 \def\ZREF@ifrefcontainsprop#1#2#3{%
493   \expandafter\ifx\expandafter\ZREF@novalue
494   #1#3#2\ZREF@novalue\ZREF@nil\@empty
495     \expandafter\@secondoftwo
496   \else
497     \expandafter\@firstoftwo
498   \fi
499 }
500 \def\ZREF@novalue{\ZREF@NOVALUE}
```

`\zref@extractdefault` The basic extracting macro is `\zref@extractdefault` with the reference name in #1, the property in #2 and the default value in #3 in case for problems.

```
501 \def\zref@extractdefault#1#2#3{%
502   \zref@ifrefundefined{#1}{%
503     \ZREF@unexpanded{#3}%
504   }{%
505     \expandafter\expandafter\expandafter\ZREF@unexpanded
506     \expandafter\expandafter\expandafter{%
```



```

507     \csname Z@E@#2\expandafter\expandafter\expandafter\endcsname
508     \csname Z@R@#1\expandafter\endcsname
509     \csname#2\endcsname{#3}\ZREF@nil
510   }%
511 }%
512 }

```

`\zref@wrapper@unexpanded`

```

513 \long\def\zref@wrapper@unexpanded#1{%
514   \let\ZREF@unexpanded\etex@unexpanded
515   #1%
516   \let\ZREF@unexpanded\@firstofone
517 }
518 \let\ZREF@unexpanded\@firstofone

```

## 6.2.10 Compatibility with babel

`\zref@wrapper@babel`

```

519 \long\def\zref@wrapper@babel#1#2{%
520   \ifcsname if@safe@actives\endcsname
521     \expandafter\@firstofone
522   \else
523     \expandafter\@secondoftwo
524   \fi
525   {%
526     \if@safe@actives
527       \expandafter\@secondoftwo
528     \else
529       \expandafter\@firstoftwo
530     \fi
531     {%
532       \begingroup
533         \csname @safe@activestruel\endcsname
534         \edef\x{#2}%
535         \expandafter\endgroup
536         \expandafter\ZREF@wrapper@babel\expandafter{\x}{#1}%
537       }%
538     }{%
539       #1{#2}%
540     }%
541   }
542 \long\def\ZREF@wrapper@babel#1#2{%
543   #2{#1}%
544 }

```

## 6.2.11 Unique counter support

`\zref@require@unique` Generate the counter `zref@unique` if the counter does not already exist.

```

545 \def\zref@require@unique{%
546   \@ifundefined{c@zref@unique}{%
547     \begingroup
548       \let\@addtoreset\@gobbletwo
549       \newcounter{zref@unique}%
550     \endgroup

```

`\thezref@unique` `\thezref@unique` is used for automatically generated unique labelnames.

```

551     \renewcommand*{\thezref@unique}{%
552       zref@number\c@zref@unique
553     }%
554   }{}%
555 }

```

### 6.2.12 Setup

`\zref@setdefault` Standard L<sup>A</sup>T<sub>E</sub>X prints “??” in bold face if a reference is not known. `\zref@default` holds the text that is printed in case of unknown references and is used, if the default was not specified during the definition of the new property by `\ref@newprop`. The global default value can be set by `\zref@setdefault`.

```
556 \def\zref@setdefault#1{%
557   \def\zref@default{#1}%
558 }
```

`\zref@default` Now we initialize `\zref@default` with the same value that L<sup>A</sup>T<sub>E</sub>X uses for its undefined references.

```
559 \zref@setdefault{%
560   \nfss@text{\reset@font\bfseries ??}%
561 }
```

#### Main property list.

`\zref@setmainlist` The name of the default property list is stored in `\ZREF@mainlist` and can be set by `\zref@setmainlist`.

```
562 \def\zref@setmainlist#1{%
563   \def\ZREF@mainlist{#1}%
564 }
565 \zref@setmainlist{main}
```

Now we create the list.

```
566 \zref@newlist\ZREF@mainlist
```

**Main properties.** The two properties `default` and `page` are created and added to the main property list. They store the data that standard L<sup>A</sup>T<sub>E</sub>X uses in its references created by `\label`.

`default` the apperance of the latest counter that is incremented by `\refstepcounter`

`page` the apperance of the page counter

```
567 \zref@newprop{default}{\@currentlabel}
568 \zref@newprop*{page}{\thepage}
569 \zref@addprop\ZREF@mainlist{default}
570 \zref@addprop\ZREF@mainlist{page}
```

#### Mark successful loading

```
571 \let\ZREF@baseok\@empty
572 \</base>
```

## 6.3 Module user

```
573 <*user>
574 \NeedsTeXFormat{LaTeX2e}
575 \ProvidesPackage{zref-user}%
576 [2009/12/08 v2.7 Module user for zref (HO)]%
577 \RequirePackage{zref-base}[2009/12/08]
578 \@ifundefined{ZREF@baseok}{\endinput}{}
```

Module user enables a small user interface. All macros are prefixed by `\z`.

First we define the pendants to the standard L<sup>A</sup>T<sub>E</sub>X referencing commands `\label`, `\ref`, and `\pageref`.

`\zlabel` Similar to `\label` the macro `\zlabel` writes a reference entry in the `.aux` file. The main property list is used. Also we add the babel patch. The `\label` command can also be used inside section titles, but it must not go into the table of contents. Therefore we have to check this situation.

```

579 \newcommand*\zlabel{%
580   \ifx\label\@gobble
581     \expandafter\@gobble
582   \else
583     \expandafter\zref@wrapper@babel\expandafter\zref@label
584   \fi
585 }%

```

**\zref** Macro **\zref** is the corresponding macro for **\ref**. Also it provides an optional argument in order to select another property.

```

586 \newcommand*{\zref}[2][default]{%
587   \zref@propexists{#1}{%
588     \zref@wrapper@babel\ZREF@zref{#2}{#1}%
589   }%
590 }%
591 \def\ZREF@zref#1{%
592   \zref@refused{#1}%
593   \zref@extract{#1}%
594 }%

```

**\zpageref** For macro **\zpageref** we just call **\zref** with property **page**.

```

595 \newcommand*\zpageref{%
596   \zref[page]%
597 }%

```

**\zrefused** For the following expandible user macros **\zrefused** should be used to notify L<sup>A</sup>T<sub>E</sub>X in case of undefined references.

```

598 \newcommand*{\zrefused}{\zref@refused}%
599 </user>

```

## 6.4 Module **abspage**

```

600 <*abspage>
601 \NeedsTeXFormat{LaTeX2e}
602 \ProvidesPackage{zref-abspage}%
603 [2009/12/08 v2.7 Module abspage for zref (H0)]%
604 \RequirePackage{zref-base}[2009/12/08]
605 \ifundefined{ZREF@baseok}{\endinput}{}

```

Module **abspage** adds a new property **abspage** to the main property list for absolute page numbers. These are recorded by the help of package **atbegshi**.

```

606 \RequirePackage{atbegshi}%

```

The counter **abspage** must not go in the clear list of **@ckpt** that is used to set counters in **.aux** files of included T<sub>E</sub>X files.

```

607 \begingroup
608   \let\@addtoreset\@gobbletwo
609   \newcounter{abspage}%
610 \endgroup
611 \setcounter{abspage}{0}%
612 \AtBeginShipout{%
613   \stepcounter{abspage}%
614 }%
615 \zref@newprop*{abspage}[0]{\the\c@abspage}%
616 \zref@addprop\ZREF@mainlist{abspage}%

```

Note that counter **abspage** shows the previous page during page processing. Before shipout the counter is incremented. Thus the property is correctly written with deferred writing. If the counter is written using **\zref@wrapper@immediate**, then the number is too small by one.

```

617 </abspage>

```

## 6.5 Module counter

```
618 <*counter>
619 \NeedsTeXFormat{LaTeX2e}
620 \ProvidesPackage{zref-counter}%
621 [2009/12/08 v2.7 Module counter for zref (HO)]%
622 \RequirePackage{zref-base}[2009/12/08]
623 \@ifundefined{ZREF@baseok}{\endinput}{}
```

For features such as `hyperref`'s `\autoref` we need the name of the counter. The property `counter` is defined and added to the main property list.

```
624 \zref@newprop{counter}{}
625 \zref@addprop{ZREF@mainlist}{counter}
```

`\refstepcounter` is the central macro where we know which counter is responsible for the reference.

```
626 \AtBeginDocument{%
627   \ZREF@patch{refstepcounter}{%
628     \def\refstepcounter#1{%
629       \zref@setcurrent{counter}{#1}%
630       \ZREF@org@refstepcounter{#1}%
631     }%
632   }%
633 }
634 </counter>
```

## 6.6 Module lastpage

```
635 <*lastpage>
636 \NeedsTeXFormat{LaTeX2e}
637 \ProvidesPackage{zref-lastpage}%
638 [2009/12/08 v2.7 Module lastpage for zref (HO)]%
639 \RequirePackage{zref-base}[2009/12/08]
640 \RequirePackage{atveryend}[2009/12/07]
641 \@ifundefined{ZREF@baseok}{\endinput}{}
```

The module `lastpage` implements the service of package `lastpage` by setting a reference `LastPage` at the end of the document. If module `abspage` is given, also the absolute page number is available, because the properties of the main property list are used.

```
642 \zref@newlist{LastPage}
643 \AfterLastShipout{%
644   \if@filesw
645     \begingroup
646       \advance\c@page\m@ne
647       \toks@\expandafter\expandafter\expandafter{%
648         \expandafter\Z@L@main
649         \Z@L@LastPage
650       }%
651       \expandafter\zref@wrapper@immediate\expandafter{%
652         \expandafter\ZREF@label\expandafter{\the\toks@}{LastPage}%
653       }%
654     \endgroup
655   \fi
656 }
657 </lastpage>
```

## 6.7 Module totpages

```
658 <*totpages>
659 \NeedsTeXFormat{LaTeX2e}
660 \ProvidesPackage{zref-totpages}%
661 [2009/12/08 v2.7 Module totpages for zref (HO)]%
662 \RequirePackage{zref-base}[2009/12/08]
663 \@ifundefined{ZREF@baseok}{\endinput}{}
```

The absolute page number of the last page is the total page number.

```
664 \RequirePackage{zref-abspage}[2009/12/08]
665 \RequirePackage{zref-lastpage}[2009/12/08]
```

`\ztotpages` Macro `\ztotpages` contains the number of pages. It can be used inside expandable calculations. It expands to zero if the reference is not yet available.

```
666 \newcommand*{\ztotpages}{%
667   \zref@extractdefault{LastPage}{abspage}{0}%
668 }
```

Also we mark the reference `LastPage` as used:

```
669 \AtBeginDocument{%
670   \zref@refused{LastPage}%
671 }
672 </totpages>
```

## 6.8 Module runs

This module does not use the label-reference-system. The reference changes with each L<sup>A</sup>T<sub>E</sub>X run and would force a rerun warning always.

```
673 <*runs>
674 \NeedsTeXFormat{LaTeX2e}
675 \ProvidesPackage{zref-runs}%
676   [2009/12/08 v2.7 Module runs for zref (H0)]%
```

`\zruns`

```
677 \providecommand*{\zruns}{0}%
678 \AtBeginDocument{%
679   \edef\zruns{\number\numexpr\zruns+1}%
680   \begingroup
681     \def\on@line{}%
682     \PackageInfo{zref-runs}{LaTeX runs: \zruns}%
683     \if@filesw
684       \immediate\write\@mainaux{%
685         \string\gdef\string\zruns{\zruns}%
686       }%
687     \fi
688   \endgroup
689 }
690 </runs>
```

## 6.9 Module perpage

```
691 <*perpage>
692 \NeedsTeXFormat{LaTeX2e}
693 \ProvidesPackage{zref-perpage}%
694   [2009/12/08 v2.7 Module perpage for zref (H0)]%
695 \RequirePackage{zref-base}[2009/12/08]
696 \@ifundefined{ZREF@baseok}{\endinput}{}%
```

This module resets a counter at page boundaries. Because of the asynchronous output routine page counter properties cannot be asked directly, references are necessary.

For detecting changed pages module `abspage` is loaded.

```
697 \RequirePackage{zref-abspage}[2009/12/08]
```

We group the properties for the needed references in the property list `perpage`. The property `pagevalue` records the correct value of the page counter.

```
698 \zref@newprop*{pagevalue}[0]{\number\c@page}
699 \zref@newlist{perpage}
700 \zref@addprop{perpage}{abspage}
```

```

701 \zref@addprop{perpage}{page}
702 \zref@addprop{perpage}{pagevalue}

```

The page value, known by the reference mechanism, will be stored in counter `zpage`.

```
703 \newcounter{zpage}
```

Counter `zref@unique` helps in generating unique reference names.

```
704 \zref@require@unique
```

In order to be able to reset the counter, we hook here into `\stepcounter`. In fact two nested hooks are used to allow other packages to use the first hook at the beginning of `\stepcounter`.

```

705 \let\ZREF@org@stepcounter\stepcounter
706 \def\stepcounter#1{%
707   \ifcsname @stepcounterhook@#1\endcsname
708     \csname @stepcounterhook@#1\endcsname
709   \fi
710   \ZREF@org@stepcounter{#1}%
711 }

```

`\zmakeperpage` Makro `\zmakeperpage` resets a counter at each page break. It uses the same syntax and semantics as `\MakePerPage` from package `perpage` [5]. The initial start value can be given by the optional argument. Default is one that means after the first `\stepcounter` on a new page the counter starts with one.

```

712 \newcommand*{\zmakeperpage}{%
713   \ifnextchar[\ZREF@makeperpage@opt{\ZREF@@makeperpage[\z@]}%
714 }

```

We hook before the counter is incremented in `\stepcounter`, package `perpage` afterwards. Thus a little calculation is necessary.

```

715 \def\ZREF@makeperpage@opt[#1]{%
716   \begingroup
717     \edef\x{\endgroup
718       \noexpand\ZREF@@makeperpage[\number\numexpr#1-1\relax]%
719     }%
720   \x
721 }

722 \def\ZREF@@makeperpage[#1]#2{%
723   \@ifundefined{@stepcounterhook@#2}{%
724     \expandafter\gdef\csname @stepcounterhook@#2\endcsname{%
725   }%
726   \expandafter\gdef\csname ZREF@perpage@#2\endcsname{%
727     \ZREF@@perpage@step{#2}{#1}%
728   }%
729   \expandafter\g@addto@macro\csname @stepcounterhook@#2\endcsname{%
730     \ifcsname ZREF@perpage@#2\endcsname
731       \csname ZREF@perpage@#2\endcsname
732     \fi
733   }%
734 }

```

`\ZREF@@perpage@step` The heart of this module follows.

```
735 \def\ZREF@@perpage@step#1#2{%
```

First the reference is generated.

```

736   \global\advance\c@zref@unique\@ne
737   \begingroup
738     \expandafter\zref@labelbylist\expandafter{\thezref@unique}{perpage}%

```

The `\expandafter` commands are necessary, because `\ZREF@temp` is also used inside of `\zref@labelbylist`.

The evaluation of the reference follows. If the reference is not yet known, we use the page counter as approximation.

```

739 \zref@ifrefundefined\thezref@unique{%
740   \global\c@zpage=\c@page
741   \global\let\thezpage\thepage
742   \expandafter\xdef\csname ZREF@abspage@#1\endcsname{\number\c@abspage}%
743 }{%

```

The reference is used to set `\thezpage` and counter `zpage`.

```

744   \global\c@zpage=\zref@extract\thezref@unique{pagevalue}\relax
745   \xdef\thezpage{\noexpand\zref@extract{\thezref@unique}{page}}%
746   \expandafter\xdef\csname ZREF@abspage@#1\endcsname{%
747     \zref@extractdefault\thezref@unique{abspage}{\number\c@abspage}%
748   }%
749 }%

```

Page changes are detected by a changed absolute page number.

```

750 \expandafter\ifx\csname ZREF@abspage@#1\expandafter\endcsname
751   \csname ZREF@currentabspage@#1\endcsname
752 \else
753   \global\csname c@#1\endcsname=#2\relax
754   \global\expandafter\let
755     \csname ZREF@currentabspage@#1\expandafter\endcsname
756     \csname ZREF@abspage@#1\endcsname
757 \fi
758 \endgroup
759 }

```

`\zunmakeperpage` Macro `\zunmakeperpage` cancels the effect of `\zmakeperpage`.

```

760 \newcommand*{\zunmakeperpage}[1]{%
761   \global\expandafter\let\csname ZREF@perpage@#1\endcsname\@undefined
762 }
763 </perpage>

```

## 6.10 Module `titleref`

```

764 <*titleref>
765 \NeedsTeXFormat{LaTeX2e}
766 \ProvidesPackage{zref-titleref}%
767 [2009/12/08 v2.7 Module titleref for zref (HO)]%
768 \RequirePackage{zref-base}[2009/12/08]
769 \@ifundefined{ZREF@baseok}{\endinput}{}
770 \RequirePackage{getttitlestring}[2009/12/08]

```

### 6.10.1 Implementation

```

771 \RequirePackage{keyval}

```

This module makes section and caption titles available for the reference system. It uses some of the ideas of package `nameref` and `titleref`.

`\zref@titleref@current` Later we will redefine the section and caption macros to catch the current title and remember the value in `\zref@titleref@current`.

```

772 \let\zref@titleref@current\@empty

```

Now we can add the property `title` is added to the main property list.

```

773 \zref@newprop{title}{\zref@titleref@current}%
774 \zref@addprop{ZREF@mainlist}{title}%

```

The title strings go into the `.aux` file, thus they need some kind of protection. Package `titleref` uses a protected expansion method. The advantage is that this can be used to cleanup the string and to remove `\label`, `\index` and other macros unwanted for referencing. But there is the risk that fragile stuff can break.

Therefore package `nameref` does not expand the string. Thus the entries can safely be written to the `.aux` file. But potentially dangerous macros such as `\label` remain in the string and can cause problems when using the string in references.

<code>\ifzref@titleref@expand</code>	<p>The switch <code>\ifzref@titleref@expand</code> distinguishes between the both methods. Package <code>nameref</code>'s behaviour is achieved by setting the switch to false, otherwise <code>titleref</code>'s expansion is used. Default is false.</p> <pre>775 \newif\ifzref@titleref@expand</pre>
<code>\ZREF@titleref@hook</code>	<p>The hook <code>\ZREF@titleref@hook</code> allows to extend the cleanup for the expansion method. Thus unnecessary macros can be removed or dangerous commands removed. The hook is executed before the expansion of <code>\zref@titleref@current</code>.</p> <pre>776 \let\ZREF@titleref@hook\@empty</pre>
<code>\zref@titleref@cleanup</code>	<p>The hook should not be used directly, instead we provide the macro <code>\zref@titleref@cleanup</code> to add stuff to the hook and prevents that a previous non-empty content is not discarded accidentally.</p> <pre> 777 \def\zref@titleref@cleanup#1{% 778   \begingroup 779   \toks@\expandafter{% 780     \ZREF@titleref@hook 781     #1% 782   }% 783   \expandafter\endgroup 784   \expandafter\def\expandafter\ZREF@titleref@hook\expandafter{% 785     \the\toks@ 786   }% 787 }</pre>
<code>\ifzref@titleref@stripperiod</code>	<p>Sometimes a title contains a period at the end. Package <code>nameref</code> removes this. This behaviour is controlled by the switch <code>\ifzref@titleref@stripperiod</code> and works regardless of the setting of option <code>expand</code>. Period stripping is the default.</p> <pre> 788 \newif\ifzref@titleref@stripperiod 789 \zref@titleref@stripperiodtrue</pre>
<code>\zref@titleref@setcurrent</code>	<p>Macro <code>\zref@titleref@setcurrent</code> sets a new current title stored in <code>\zref@titleref@current</code>. Some cleanup and expansion is performed that can be controlled by the previous switches.</p> <pre> 790 \def\zref@titleref@setcurrent#1{% 791   \ifzref@titleref@expand 792     \GetTitleStringExpand{#1}% 793   \else 794     \GetTitleStringNonExpand{#1}% 795   \fi 796   \edef\zref@titleref@current{% 797     \detokenize\expandafter{\GetTitleStringResult}% 798   }% 799   \ifzref@titleref@stripperiod 800     \edef\zref@titleref@current{% 801       \expandafter\ZREF@stripperiod\zref@titleref@current 802       \@empty.\@empty\@nil 803     }% 804   \fi 805 }% 806 \GetTitleStringDisableCommands{% 807   \ZREF@titleref@hook 808 }</pre>
<code>\ZREF@stripperiod</code>	<p>If <code>\ZREF@stripperiod</code> is called, the argument consists of space tokens and tokens with catcode 12 (other), because of <math>\varepsilon</math>-<math>\text{\TeX}</math>'s <code>\detokenize</code>.</p> <pre>809 \def\ZREF@stripperiod#1.\@empty#2\@nil{#1}%</pre>



### 6.10.2 User interface

`\ztitlerefsetup` The behaviour of module `titleref` is controlled by switches and a hook. They can be set by `\ztitlerefsetup` with a key value interface, provided by package `keyval`. Also the current title can be given explicitly by the key `title`.

```

810 \define@key{ZREF@TR}{expand}[true]{%
811   \csname zref@titleref@expand#1\endcsname
812 }%
813 \define@key{ZREF@TR}{stripperperiod}[true]{%
814   \csname zref@titleref@stripperperiod#1\endcsname
815 }%
816 \define@key{ZREF@TR}{cleanup}{%
817   \zref@titleref@cleanup{#1}%
818 }%
819 \define@key{ZREF@TR}{title}{%
820   \def\zref@titleref@current{#1}%
821 }%
822 \newcommand*{\ztitlerefsetup}{%
823   \setkeys{ZREF@TR}%
824 }%
```

`\ztitleref` The user command `\ztitleref` references the title. For safety `\label` is disabled to prevent multiply defined references.

```

825 \newcommand*{\ztitleref}{%
826   \zref@wrapper@babel\ZREF@titleref
827 }%
828 \def\ZREF@titleref#1{%
829   \begingroup
830     \zref@refused{#1}%
831     \let\label\gobble
832     \zref@extract{#1}{title}%
833   \endgroup
834 }%
```

### 6.10.3 Patches for section and caption commands

The section and caption macros are patched to extract the title data.

Captions of figures and tables.

```

835 \AtBeginDocument{%
836   \ZREF@patch{@caption}{%
837     \long\def\@caption#1[#2]{%
838       \zref@titleref@setcurrent{#2}%
839       \ZREF@org@@caption{#1}[{#2}]%
840     }%
841   }%
```

Section commands without star. The title version for the table of contents is used because it is usually shorter and more robust.

```

842   \ZREF@patch{@part}{%
843     \def\@part[#1]{%
844       \zref@titleref@setcurrent{#1}%
845       \ZREF@org@@part[{#1}]%
846     }%
847   }%
848   \ZREF@patch{@chapter}{%
849     \def\@chapter[#1]{%
850       \zref@titleref@setcurrent{#1}%
851       \ZREF@org@@chapter[{#1}]%
852     }%
853   }%
854   \ZREF@patch{@sect}{%
855     \def\@sect#1#2#3#4#5#6[#7]{%
```

```

856     \zref@titleref@setcurrent{#7}%
857     \ZREF@org@ssect{#1}{#2}{#3}{#4}{#5}{#6} [{#7}]%
858 }%
859 }%

```

The star versions of the section commands.

```

860 \ZREF@patch{@spart}{%
861   \def\@spart#1{%
862     \zref@titleref@setcurrent{#1}%
863     \ZREF@org@spart{#1}%
864   }%
865 }%
866 \ZREF@patch{@schapter}{%
867   \def\@schapter#1{%
868     \zref@titleref@setcurrent{#1}%
869     \ZREF@org@schapter{#1}%
870   }%
871 }%
872 \ZREF@patch{@ssect}{%
873   \def\@ssect#1#2#3#4#5{%
874     \zref@titleref@setcurrent{#5}%
875     \ZREF@org@ssect{#1}{#2}{#3}{#4}{#5}%
876   }%
877 }%

```

Class beamer.

```

878 \@ifclassloaded{beamer}{%
879   \ZREF@patch{beamer@section}{%
880     \long\def\beamer@section[#1]{%
881       \zref@titleref@setcurrent{#1}%
882       \ZREF@org@beamer@section[{#1}]%
883     }%
884   }%
885   \ZREF@patch{beamer@subsection}{%
886     \long\def\beamer@subsection[#1]{%
887       \zref@titleref@setcurrent{#1}%
888       \ZREF@org@beamer@subsection[{#1}]%
889     }%
890   }%
891   \ZREF@patch{beamer@subsubsection}{%
892     \long\def\beamer@subsubsection[#1]{%
893       \zref@titleref@setcurrent{#1}%
894       \ZREF@org@beamer@subsubsection[{#1}]%
895     }%
896   }%
897 }{}%

```

Package titlesec.

```

898 \@ifpackageloaded{titlesec}{%
899   \ZREF@patch{ttl@ssect@i}{%
900     \def\ttl@ssect@i#1#2[#3]#4{%
901       \zref@titleref@setcurrent{#4}%
902       \ZREF@org@ttl@ssect@i{#1}{#2} [{#3}] {#4}%
903     }%
904   }%
905 }{}%

```

Package longtable: some support for its \caption. However \label inside the caption is not supported.

```

906 \@ifpackageloaded{longtable}{%
907   \ZREF@patch{LT@c@ption}{%
908     \def\LT@c@ption#1[#2]#3{%
909       \ZREF@org@LT@c@ption{#1} [{#2}] {#3}%
910       \zref@titleref@setcurrent{#2}%

```

```

911     }%
912   }%
913 }{}%

Package listings: support for its caption.
914 \@ifpackageloaded{listings}{%
915   \ZREF@patch{lst@MakeCaption}{%
916     \def\lst@MakeCaption{%
917       \ifx\lst@label\@empty
918       \else
919         \expandafter\zref@titleref@setcurrent\expandafter{%
920           \lst@caption
921         }%
922       \fi
923     \ZREF@org@lst@MakeCaption
924   }%
925 }%
926 }{}%
927 }
928 \</titleref>

```

## 6.11 Module xr

```

929 <*xr>
930 \NeedsTeXFormat{LaTeX2e}
931 \ProvidesPackage{zref-xr}%
932 [2009/12/08 v2.7 Module xr for zref (H0)]%
933 \RequirePackage{zref-base}[2009/12/08]
934 \@ifundefined{ZREF@baseok}{\endinput}{}
935 \RequirePackage{keyval}

```

We declare property `url`, because this is added, if a reference is imported and has not already set this field. Or if `hyperref` is used, then this property can be asked.

```

936 \zref@newprop{url}{}%

```

Most code, especially the handling of the `.aux` files are taken from David Carlisle's `xr` package. Therefore I drop the documentation for these macros here.

`\zref@xr@ext` If the URL is not specied, then assume processed file with a guessed extension. Use the setting of `hyperref` if available.

```

937 \providecommand*{\zref@xr@ext}{}%
938 \@ifundefined{XR@ext}{pdf}{\XR@ext}%
939 }%

```

`\ifZREF@xr@zreflabel` The use of the star form of `\externaldocument` is remembered in the switch `\ifZREF@xr@zreflabel`.

```

940 \newif\ifZREF@xr@zreflabel

```

`\externaldocument` In its star form it looks for `\newlabel`, otherwise for `\zref@newlabel`. Later we will read `.aux` files that expects `@` to have catcode 11 (letter).

```

941 \newcommand*{\externaldocument}{%
942   \begingroup
943     \csname @safe@actives@true\endcsname
944     \makeatletter
945     \@ifstar{%
946       \ZREF@xr@zreflabelfalse
947       \@testopt\ZREF@xr@externaldocument{}%
948     }{%
949       \ZREF@xr@zreflabeltrue
950       \@testopt\ZREF@xr@externaldocument{}%
951     }%
952 }%

```

If the `\include` featurer was used, there can be several `.aux` files. These files are read one after another, especially they are not recursively read in order to save read registers. Thus it can happen that the read order of the `newlabel` commands differs from L<sup>A</sup>T<sub>E</sub>X's order using `\input`.

`\ZREF@xr@externaldocument` It reads the remaining arguments. `\newcommand` comes in handy for the optional argument.

```

953 \def\ZREF@xr@externaldocument[#1]#2{%
954     \def\ZREF@xr@prefix{#1}%
955     \let\ZREF@xr@filelist\@empty
956     \edef\ZREF@xr@file{#2.aux}%
957     \filename@parse{#2}%
958     \@testopt\ZREF@xr@graburl{#2.\zref@xr@ext}%
959 }%
960 \def\ZREF@xr@graburl[#1]{%
961     \edef\ZREF@xr@url{#1}%
962     \ZREF@xr@checkfile
963 \endgroup
964 }%
```

`\ZREF@xr@processfile` We follow `xr` here, `\IfFileExists` offers a nicer test, but we have to open the file anyway.

```

965 \def\ZREF@xr@checkfile{%
966     \openin\@inputcheck\ZREF@xr@file\relax
967     \ifeof\@inputcheck
968         \PackageWarning{zref-xr}{%
969             File '\ZREF@xr@file' not found or empty,\MessageBreak
970             labels not imported%
971         }%
972     \else
973         \PackageInfo{zref-xr}{%
974             Label \ifZREF@xr@zreflabel (zref) \fi import from '\ZREF@xr@file'%
975         }%
976         \def\ZREF@xr@found{0}%
977         \def\ZREF@xr@ignored{0}%
978         \ZREF@xr@processfile
979         \closein\@inputcheck
980         \begingroup
981             \let\on@line\@empty
982             \PackageInfo{zref-xr}{%
983                 Statistics for '\ZREF@xr@file': %
984                 \ZREF@xr@found\space found, %
985                 \ZREF@xr@ignored\space ignored%
986             }%
987         \endgroup
988     \fi
989     \ifx\ZREF@xr@filelist\@empty
990     \else
991         \edef\ZREF@xr@file{\expandafter\@car\ZREF@xr@filelist\@nil}%
992         \edef\ZREF@xr@filelist{\expandafter\@cdr\ZREF@xr@filelist\@nil}%
993         \expandafter\ZREF@xr@checkfile
994     \fi
995 }%
```

`\ZREF@xr@processfile`

```

996 \def\ZREF@xr@processfile{%
997     \read\@inputcheck to\ZREF@xr@line
998     \expandafter\ZREF@xr@processline\ZREF@xr@line..\ZREF@nil
999     \ifeof\@inputcheck
1000 \else
1001     \expandafter\ZREF@xr@procesfile
1002 \fi
```

```

1003 }%

\ZREF@xr@processline The most work must be done for analyzing the arguments of \newlabel.

1004 \long\def\ZREF@xr@processline#1#2#3\ZREF@nil{%
1005   \def\x{#1}%
1006   \toks@{#2}%
1007   \ifZREF@xr@zreflabel
1008     \ifx\x\ZREF@xr@zref@newlabel
1009       \expandafter\ZREF@xr@process@zreflabel\ZREF@xr@line...\ZREF@nil
1010     \fi
1011   \else
1012     \ifx\x\ZREF@xr@newlabel
1013       \expandafter\ZREF@xr@process@label\ZREF@xr@line...[]\ZREF@nil
1014     \fi
1015   \fi
1016   \ifx\x\ZREF@xr@@input
1017     \edef\ZREF@xr@filelist{%
1018       \etex@unexpanded\expandafter{\ZREF@xr@filelist}%
1019       {\filename@area\the\toks@}%
1020     }%
1021   \fi
1022   \ifeof\@inputcheck
1023   \else
1024     \expandafter\ZREF@xr@processfile
1025   \fi
1026 }%
1027 \def\ZREF@xr@process@zreflabel\zref@newlabel#1#2#3\ZREF@nil{%
1028   \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1029   \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1030   \def\x{#2}%
1031   \@ifundefined{\ZREF@xr@refname}{%
1032     \let\ZREF@xr@list\x
1033     \ifx\ZREF@xr@list\@empty
1034       \PackageWarningNoLine{zref-xr}{%
1035         Label ‘#1’ without properties ignored\MessageBreak
1036         in file ‘\ZREF@xr@file’%
1037       }%
1038       \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
1039     \else
1040       \expandafter\ZREF@xr@checklist\x\ZREF@nil
1041       \expandafter\global\expandafter\let
1042         \csname \ZREF@xr@refname\endcsname\x
1043     \fi
1044     \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1045   }{%
1046     \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
1047   }%
1048 }%
1049 \def\ZREF@xr@process@label\newlabel#1#2#3[#4]#5\ZREF@nil{%
1050   \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1051   \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1052   \def\x{#2}%
1053   \@ifundefined{\ZREF@xr@refname}{%
1054     \expandafter\ZREF@xr@scanparams
1055       \csname\ZREF@xr@refname\expandafter\endcsname
1056       \x{}{}{}{}{}\ZREF@nil
1057   \ifx\\#4\\%
1058   \else
1059     % ntheorem knows an optional argument at the end of \newlabel
1060     \zref@ifpropundefined{theotype}{%
1061       \zref@newprop{theotype}{}%
1062     }{}%
1063     \expandafter\g@addto@macro

```

```

1064      \csname\ZREF@xr@refname\endcsname{\theotype{#4}}%
1065      \fi
1066      \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1067    }{%
1068      \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
1069    }%
1070  }
1071  \def\ZREF@xr@zref@newlabel{\zref@newlabel}%
1072  \def\ZREF@xr@newlabel{\newlabel}%
1073  \def\ZREF@xr@input{\@input}%

```

\ZREF@xr@ignorewarning

```

1074  \def\ZREF@xr@ignorewarning#1{%
1075    \PackageWarningNoLine{zref-xr}{%
1076      Label ‘#1’ is already in use\MessageBreak
1077      in file ‘\ZREF@xr@file’%
1078    }%
1079    \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
1080  }%

```

\ZREF@xr@checklist

```

1081  \def\ZREF@xr@checklist#1#2#3\ZREF@nil{%
1082    \ifx\@undefined#1\relax
1083      \expandafter\ZREF@xr@checkkey\string#1\@nil
1084      \fi
1085      \ifx\#3\%
1086      \else
1087        \@ReturnAfterFi{%
1088          \ZREF@xr@checklist#3\ZREF@nil
1089        }%
1090      \fi
1091    }%
1092    \long\def\@ReturnAfterFi#1\fi{\fi#1}%
1093    \def\ZREF@xr@checkkey#1#2\@nil{%
1094      \zref@ifpropundefined{#2}{%
1095        \zref@newprop{#2}{}%
1096      }{%
1097    }%

```

\ZREF@xr@scanparams

```

1098  \def\ZREF@xr@scanparams#1#2#3#4#5#6#7\ZREF@nil{%
1099    \global\let#1\@empty
1100    \ZREF@foundfalse
1101    \ZREF@xr@scantitleref#1#2\TR@TitleReference{}{}\ZREF@nil
1102    \ifZREF@found
1103    \else
1104      \g@addto@macro#1{\default{#2}}%
1105      \fi
1106      % page
1107      \g@addto@macro#1{\page{#3}}%
1108      % nameref title
1109      \ifZREF@found
1110      \else
1111        \ifx\#4\%
1112        \else
1113          \zref@ifpropundefined{title}{%
1114            \zref@newprop{title}{}%
1115          }{%
1116            \g@addto@macro#1{\title{#4}}%
1117          \fi
1118        \fi
1119        % anchor

```

```

1120 \ifx\\#5\\%
1121 \else
1122   \zref@ifpropundefined{anchor}{%
1123     \zref@newprop{anchor}{}%
1124   }{}%
1125   \g@addto@macro#1{\anchor{#5}}%
1126 \fi
1127 \ifx\\#6\\%
1128 \else
1129   \zref@ifpropundefined{url}{%
1130     \zref@newprop{url}{}%
1131   }{}%
1132   \g@addto@macro#1{\url{#6}}%
1133 \fi
1134 }%

```

\ZREF@xr@scantitleref

```

1135 \def\ZREF@xr@scantitleref#1#2\TR@TitleReference#3#4#5\ZREF@nil{%
1136 \ifx\\#5\\%
1137 \else
1138   \g@addto@macro#1{%
1139     \default{#3}%
1140     \title{#4}%
1141   }%
1142   \ZREF@foundtrue
1143 \fi
1144 }%

```

\ZREF@xr@urlcheck

```

1145 \def\ZREF@xr@urlcheck#1{%
1146 \zref@ifrefcontainsprop{#1}{anchor}{%
1147   \zref@ifrefcontainsprop{#1}{url}{%
1148     }{%
1149       \expandafter\g@addto@macro\csname Z@R@#1\expandafter\endcsname
1150       \expandafter{%
1151         \expandafter\url\expandafter{\ZREF@xr@url}%
1152       }%
1153     }%
1154   }{%
1155   }%
1156 }%

```

\zxrsetup Just one key for setting the default extension is currently used.

```

1157 \define@key{ZREF@XR}{ext}{%
1158 \def\zref@xr@ext{#1}%
1159 }%
1160 \newcommand*{\zxrsetup}{%
1161 \setkeys{ZREF@XR}%
1162 }%
1163 \</xr>

```

## 6.12 Module hyperref

UNFINISHED :-(

```

1164 \<*hyperref>
1165 \NeedsTeXFormat{LaTeX2e}
1166 \ProvidesPackage{zref-hyperref}%
1167 [2009/12/08 v2.7 Module hyperref for zref (HO)]%
1168 \RequirePackage{zref-base}[2009/12/08]
1169 \@ifundefined{ZREF@baseok}{\endinput}{}

```

```

1170 \zref@newprop{anchor}[] {%
1171   \@ifundefined{@currentHref}{\@currentHref}%
1172 }%
1173 \zref@addprop\ZREF@mainlist{anchor}%
1174 </hyperref>

```

## 6.13 Module **savepos**

Module **savepos** provides an interface for pdfTeX's `\pdfsavepos`, see the manual for pdfTeX.

### 6.13.1 Identification

```

1175 <*savepos>
1176 \NeedsTeXFormat{LaTeX2e}
1177 \ProvidesPackage{zref-savepos}%
1178   [2009/12/08 v2.7 Module savepos for zref (H0)]%
1179 \RequirePackage{zref-base}[2009/12/08]
1180 \@ifundefined{ZREF@baseok}{\endinput}{}

```

### 6.13.2 Availability

First we check, whether the feature is available.

```

1181 \begingroup
1182   \@ifundefined{pdfsavepos}{%
1183     \ZREF@ErrorNoLine{%
1184       \string\pdfsavepos\space is not supported.\MessageBreak
1185       It is provided by pdfTeX (1.40) or XeTeX%
1186     }\ZREF@UpdatePdfTeX
1187   \endgroup
1188   \endinput
1189 }{}%
1190 \endgroup

```

In PDF mode we are done. However support for DVI mode was added later in version 1.40.0. In earlier versions `\pdfsavepos` is defined, but its execution raises an error. Note that XeTeX also provides `\pdfsavepos`.

```

1191 \RequirePackage{ifpdf}
1192 \ifpdf
1193 \else
1194   \begingroup\expandafter\expandafter\expandafter\endgroup
1195   \expandafter\ifx\csname pdftexversion\endcsname\relax
1196   \else
1197     \ifnum\pdftexversion<140 %
1198       \ZREF@ErrorNoLine{%
1199         \string\pdfsavepos\space is not supported in DVI mode\MessageBreak
1200         of this pdfTeX version%
1201       }\ZREF@UpdatePdfTeX
1202     \expandafter\expandafter\expandafter\endinput
1203   \fi
1204 \fi
1205 \fi

```

### 6.13.3 Setup

```

1206 \zref@newlist{savepos}
1207 \zref@newprop*{posx}[0]{\the\pdflastxpos}
1208 \zref@newprop*{posy}[0]{\the\pdflastypos}
1209 \zref@addprop{savepos}{posx}
1210 \zref@addprop{savepos}{posy}

```

### 6.13.4 User macros

`\zsavepos` The current location is stored in a reference with the given name.



```

1211 \def\zsavepos#1{%
1212   \@bsphack
1213   \if@filesw
1214     \pdfsavepos
1215     \zref@labelbylist{#1}{savepos}%
1216   \fi
1217   \@esphack
1218 }

```

`\zposx` The horizontal and vertical position are available by `\zposx` and `\zposy`. Do not rely on absolute positions. They differ in DVI and PDF mode of pdfTeX. Use differences instead. The unit of the position numbers is sp.

```

1219 \newcommand*{\zposx}[1]{%
1220   \zref@extract{#1}{posx}%
1221 }%
1222 \newcommand*{\zposy}[1]{%
1223   \zref@extract{#1}{posy}%
1224 }%

```

Typically horizontal and vertical positions are used inside calculations. Therefore the extracting macros should be expandable and babel's patch is not applicable.

Also it is in the responsibility of the user to marked used positions by `\zrefused` in order to notify L<sup>A</sup>T<sub>E</sub>X about undefined references.

```

1225 </savepos>

```

## 6.14 Module dotfill

```

1226 <*dotfill>
1227 \NeedsTeXFormat{LaTeX2e}
1228 \ProvidesPackage{zref-dotfill}%
1229   [2009/12/08 v2.7 Module dotfill for zref (H0)]%
1230 \RequirePackage{zref-base}[2009/12/08]
1231 \ifundefined{ZREF@baseok}{\endinput}{\}

```

For measuring the width of `\zdotfill` we use the features provided by module `savepos`.

```

1232 \RequirePackage{zref-savepos}[2009/12/08]

```

For automatically generated label names we use the unique counter of module `base`.

```

1233 \zref@require@unique

```

Configuration is done by the key value interface of package `keyval`.

```

1234 \RequirePackage{keyval}

```

The definitions of the keys follow.

```

1235 \define@key{ZREF@DF}{unit}{%
1236   \def\ZREF@df@unit{#1}%
1237 }
1238 \define@key{ZREF@DF}{min}{%
1239   \def\ZREF@df@min{#1}%
1240 }
1241 \define@key{ZREF@DF}{dot}{%
1242   \def\ZREF@df@dot{#1}%
1243 }

```

Defaults are set, see user interface.

```

1244 \providecommand\ZREF@df@min{2}
1245 \providecommand\ZREF@df@unit{.44em}
1246 \providecommand\ZREF@df@dot{.}

```

`\zdotfillsetup` Configuration of `\zdotfill` is done by `\zdotfillsetup`.

```

1247 \newcommand*{\zdotfillsetup}{\setkeys{ZREF@DF}}

```

`\zdotfill` `\zdotfill` sets labels at the left and the right to get the horizontal position.  
`\zsavepos` is not used, because we do not need the vertical position.

```

1248 \newcommand*{\zdotfill}{%
1249   \leavevmode
1250   \global\advance\c@zref@unique\@ne
1251   \begingroup
1252     \def\ZREF@temp{zref@\number\c@zref@unique}%
1253     \pdfsavepos
1254     \zref@labelbyprops{\thezref@unique L}{posx}%
1255     \setlength{\dimen@}{\ZREF@df@unit}%
1256     \zref@ifrefundefined{\thezref@unique R}{%
1257       \ZREF@dotfill
1258     }{%
1259       \ifnum\numexpr\zposx{\thezref@unique R}-\zposx{\thezref@unique L}\relax
1260         <\dimexpr\ZREF@df@min\dimen@\relax
1261         \hfill
1262       \else
1263         \ZREF@dotfill
1264       \fi
1265     }%
1266     \pdfsavepos
1267     \zref@labelbyprops{\thezref@unique R}{posx}%
1268   \endgroup
1269   \kern\z@
1270 }
```

`\ZREF@dotfill` Help macro that actually sets the dots.

```

1271 \def\ZREF@dotfill{%
1272   \cleaders\hb@xt@\dimen@{\hss\ZREF@df@dot\hss}\hfill
1273 }

1274 </dotfill>
```

## 7 Test

### 7.1 `\zref@localaddprop`

```

1275 <*test1>
1276 \NeedsTeXFormat{LaTeX2e}
1277 \nofiles
1278 \documentclass{article}
1279 \usepackage{zref-base}[2009/12/08]
1280 \usepackage{qstest}
1281 \IncludeTests{*}
1282 \LogTests{log}{*}{*}
1283
1284 \makeatletter
1285 \begin{qstest}\localaddprop{\localaddprop}
1286   \Expect*{\Z@L@main}*{\{default\}{page}}%
1287   \zref@newprop{foobar}{F00}%
1288   \zref@newlist{alist}%
1289   \Expect*{\Z@L@alist}{}%
1290   \begingroup
1291     \zref@localaddprop{main}{foobar}%
1292     \Expect*{\Z@L@main}{\{default\}{page}{foobar}}%
1293     \zref@localaddprop{alist}{page}%
1294     \Expect*{\Z@L@alist}{\{page}}%
1295   \endgroup
1296   \Expect*{\Z@L@main}*{\{default\}{page}}%
1297   \Expect*{\Z@L@alist}{}%
1298 \end{qstest}
1299 \@@end
```

1300  $\langle$ /test1 $\rangle$

## 7.2 Module runs

```
1301  $\langle$ *test-runs $\rangle$ 
1302 \NeedsTeXFormat{LaTeX2e}
1303 \documentclass{article}
1304 \usepackage{zref-runs}[2009/12/08]
1305 \usepackage{qstest}
1306 \IncludeTests{*}
1307 \LogTests{log}{*}{*}
1308
1309 \begin{qstest}{zruns-preamble}{zruns-preamble}
1310   \Expect{0}{*}{\zruns}%
1311 \end{qstest}
1312
1313 \AtBeginDocument{%
1314   \begin{qstest}{zruns-atbegindocument}{zruns-atbegindocument}%
1315     \Expect*{\number\ExpectRuns}{*}{\zruns}%
1316   \end{qstest}%
1317 }
1318
1319 \begin{document}
1320 \begin{qstest}{zruns-document}{zruns-document}
1321   \Expect*{\number\ExpectRuns}{*}{\zruns}%
1322 \end{qstest}
1323 \end{document}
1324  $\langle$ /test-runs $\rangle$ 
```

## 8 Installation

### 8.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[CTAN:macros/latex/contrib/oberdiek/zref.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/zref.pdf](#) Documentation.

**Bundle.** All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

TDS refers to the standard “A Directory Structure for T<sub>E</sub>X Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

### 8.2 Bundle installation

**Unpacking.** Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

**Script installation.** Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

---

<sup>1</sup>[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

### 8.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain- $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ :

```
tex zref.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>zref.sty</code>	<code>→ tex/latex/oberdiek/zref.sty</code>
<code>zref-base.sty</code>	<code>→ tex/latex/oberdiek/zref-base.sty</code>
<code>zref-abspage.sty</code>	<code>→ tex/latex/oberdiek/zref-abspage.sty</code>
<code>zref-counter.sty</code>	<code>→ tex/latex/oberdiek/zref-counter.sty</code>
<code>zref-dotfill.sty</code>	<code>→ tex/latex/oberdiek/zref-dotfill.sty</code>
<code>zref-hyperref.sty</code>	<code>→ tex/latex/oberdiek/zref-hyperref.sty</code>
<code>zref-lastpage.sty</code>	<code>→ tex/latex/oberdiek/zref-lastpage.sty</code>
<code>zref-perpage.sty</code>	<code>→ tex/latex/oberdiek/zref-perpage.sty</code>
<code>zref-runs.sty</code>	<code>→ tex/latex/oberdiek/zref-runs.sty</code>
<code>zref-savepos.sty</code>	<code>→ tex/latex/oberdiek/zref-savepos.sty</code>
<code>zref-titleref.sty</code>	<code>→ tex/latex/oberdiek/zref-titleref.sty</code>
<code>zref-totpages.sty</code>	<code>→ tex/latex/oberdiek/zref-totpages.sty</code>
<code>zref-user.sty</code>	<code>→ tex/latex/oberdiek/zref-user.sty</code>
<code>zref-xr.sty</code>	<code>→ tex/latex/oberdiek/zref-xr.sty</code>
<code>zref.pdf</code>	<code>→ doc/latex/oberdiek/zref.pdf</code>
<code>zref-example.tex</code>	<code>→ doc/latex/oberdiek/zref-example.tex</code>
<code>zref-example-lastpage.tex</code>	<code>→ doc/latex/oberdiek/zref-example-lastpage.tex</code>
<code>test/zref-test1.tex</code>	<code>→ doc/latex/oberdiek/test/zref-test1.tex</code>
<code>test/zref-test-runs.tex</code>	<code>→ doc/latex/oberdiek/test/zref-test-runs.tex</code>
<code>zref.dtx</code>	<code>→ source/latex/oberdiek/zref.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

### 8.4 Refresh file name databases

If your  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  distribution (`te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , `mik $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , ...) relies on file name databases, you must refresh these. For example, `te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$`  users run `texhash` or `mktextlsr`.

### 8.5 Some details for the interested

**Attached source.** The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk zref.pdf unpack_files output .
```

**Unpacking with  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ :** Run `docstrip` and extract the files.

**$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ :** Generate the documentation.

If you insist on using  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  for `docstrip` (really, `docstrip` does not need  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{zref.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

**Generating the documentation.** You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL<sup>A</sup>T<sub>E</sub>X:

```
pdflatex zref.dtx
makeindex -s gind.ist zref.idx
pdflatex zref.dtx
makeindex -s gind.ist zref.idx
pdflatex zref.dtx
```

## 9 References

- [1] Package `footmisc`, Robin Fairbairns, 2004/01/23 v5.3a.[CTAN:macros/latex/contrib/footmisc/footmisc.dtx](#)
- [2] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/08/16 v6.75c.[CTAN:macros/latex/contrib/hyperref/](#)
- [3] Package `lastpage`, Jeff Goldberg, 1994/06/25 v0.1b.[CTAN:macros/latex/contrib/lastpage/](#)
- [4] Package `nameref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12 v2.24.[CTAN:macros/latex/contrib/hyperref/nameref.dtx](#)
- [5] Package `perpage`, David Kastrup, 2002/12/20 v1.0.[CTAN:macros/latex/contrib/bigfoot/perpage.dtx](#)
- [6] Package `titleref`, Donald Arsenau, 2001/04/05 v3.1.[CTAN:macros/latex/contrib/misc/titleref.sty](#)
- [7] Package `totpages`, Wilhelm Müller, 1999/07/14 v1.00.[CTAN:macros/latex/contrib/totpages/](#)
- [8] Package `xr`, David Carlisle, 1994/05/28 v5.02.[CTAN:macros/latex/required/tools/xr.pdf](#)
- [9] Package `xr-hyper`, David Carlisle, 2000/03/22 v6.00beta4.[CTAN:macros/latex/contrib/hyperref/xr-hyper.sty](#)

## 10 History

[2006/02/20 v1.0]

- First version.

[2006/05/03 v1.1]

- Module `perpage` added.
- Module redesign as packages.

[2006/05/25 v1.2]

- Module `dotfillmin` added.
- Module `base`: macros `\zref@require@unique` and `\thezref@unique` added (used by modules `titleref` and `dotfillmin`).

[2006/09/08 v1.3]

- Typo fixes and English cleanup by Per Starback.

[2007/01/23 v1.4]

- Typo in macro name fixed in documentation.

[2007/02/18 v1.5]

- `\zref@getcurrent` added (suggestion of Igor Akkerman).
- Module `savepos` also supports XeTeX.

[2007/04/06 v1.6]

- Fix in modules `abspage` and `base`: Now counter `abspage` and `zref@unique` are not remembered by `\include`.
- Beamer support for module `titleref`.

[2007/04/17 v1.7]

- Package `atbegshi` replaces `everyshi`.

[2007/04/22 v1.8]

- `\zref@wrapper@babel` and `\zref@refused` are now expandable if `babel` is not used or `\if@safe@actives` is already set to true. (Feature request of Josselin Noirel)

[2007/05/02 v1.9]

- Module `titleref`: Some support for `\caption` of package `longtable`, but only if `\label` is given after `\caption`.

[2007/05/06 v2.0]

- Uses package `etexcmds` for accessing  $\varepsilon$ -TeX's `\unexpanded`.

[2007/05/28 v2.1]

- Module `titleref` supports caption of package `listings`.
- Fixes in module `titleref` for support of packages `titlesec` and `longtable`.

[2008/09/21 v2.2]

- Module `base`: `\zref@iflistcontainsprop` is documented, but a broken `\zref@listcontainsprop` implemented. Name and implementation fixed (thanks Ohad Kammar).

[2008/10/01 v2.3]

- `\zref@localaddprop` added (feature request of Ohad Kammar).
- Module `lastpage`: list 'LastPage' added. Label 'LastPage' will use the properties of this list (default is empty) along with the properties of the main list.

[2009/08/07 v2.4]

- Module runs added.

[2009/12/06 v2.5]

- Module lastpage: Uses package atveryend.
- Module titleref: Further commands are disabled during string expansion, imported from package nameref.

[2009/12/07 v2.6]

- Version date added for package atveryend.

[2009/12/08 v2.7]

- Module titleref: Use of package gettitlestring.

## 11 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; numbers in roman refer to the code lines where the entry is used.

Symbols		
\@@end	1299	\@namedef 361
\@ReturnAfterFi	1087, 1092	\@ne 736, 1250
\@addtoreset	548, 608	\@newl@bel 234
\@auxout	423	\@nil 802, 809, 991, 992, 1083, 1093
\@bsphack	379, 389, 1212	\@part 843
\@caption	837	\@schapter 867
\@car	991	\@secondoftwo 252,
\@cdr	992	280, 336, 457, 482, 495, 523, 527
\@chapter	849	\@sect 855
\@currentHref	1171	\@spart 861
\@currentlabel	567	\@ssect 873
\@ehc	245, 259, 343	\@testopt 947, 950, 958
\@empty	188, 239, 351,	\@tfor 269, 284, 428
	427, 494, 571, 772, 776, 802,	\@undefined 761, 1082
	809, 917, 955, 981, 989, 1033, 1099	\\ 26, 27, 28,
\@esphack	386, 406, 1217	29, 121, 123, 125, 126, 138, 141,
\@firstofone	201, 516, 518, 521	1057, 1085, 1111, 1120, 1127, 1136
\@firstoftwo	250, 278, 334, 455, 497, 529	
\@for	393	<b>A</b>
\@gobble	197, 580, 581, 831	\AddLineBeginAux 230
\@gobbletwo	548, 608	\advance 646, 736, 1250
\@ifclassloaded	878	\AfterLastShipout 643
\@ifdefinable	238	\Alph 6
\@ifnextchar	358, 713	\anchor 1125
\@ifpackageloaded	898, 906, 914	\AtBeginDocument
\@ifstar	347, 945	626, 669, 678, 835, 1313
\@ifundefined	160, 205,	\AtBeginShipout 612
	546, 578, 605, 623, 641, 663,	\AtEndOfPackage 163
	696, 723, 769, 934, 938, 1031,	
	1053, 1169, 1171, 1180, 1182, 1231	<b>B</b>
\@input	1073	\beamer@section 880
\@inputcheck	966, 967, 979, 997, 999, 1022	\beamer@subsection 886
\@latex@warning	466	\beamer@subsubsection 892
\@mainaux	684	\begin 24, 70, 76, 124,
		140, 1285, 1309, 1314, 1319, 1320
		\bfseries 560

<b>C</b>		<b>G</b>	
<code>\c@abspage</code>	615, 742, 747	<code>\g@addto@macro</code>	301, 322, 729, 1063, 1104, 1107, 1116, 1125, 1132, 1138, 1149
<code>\c@page</code>	646, 698, 740	<code>\G@refundefinedtrue</code>	465
<code>\c@zpage</code>	740, 744	<code>\gdef</code>	367, 685, 724, 726
<code>\c@zref@unique</code>	552, 736, 1250, 1252	<code>\GetTitleStringDisableCommands</code>	806
<code>\chapter</code>	25, 31, 33, 52	<code>\GetTitleStringExpand</code>	792
<code>\ChapterPages</code>	61, 81	<code>\GetTitleStringNonExpand</code>	794
<code>\ChapterStart</code>	48, 103, 118, 134	<code>\GetTitleStringResult</code>	797
<code>\ChapterStop</code>	55, 116, 133, 152		
<code>\cleaders</code>	1272	<b>H</b>	
<code>\cleardoublepage</code>	49, 56	<code>\hb@xt@</code>	1272
<code>\closein</code>	979	<code>\hfill</code>	1261, 1272
<code>\csname</code>	196, 199, 200, 239, 249, 263, 288, 301, 314, 333, 362, 363, 367, 370, 373, 383, 430, 432, 437, 438, 454, 474, 485, 486, 488, 507, 508, 509, 533, 708, 724, 726, 729, 731, 742, 746, 750, 751, 753, 755, 756, 761, 811, 814, 943, 1042, 1055, 1064, 1149, 1195	<code>\hss</code>	1272
<code>\current@chapid</code>	50, 58		
<b>D</b>		<b>I</b>	
<code>\DeclareOption</code>	162	<code>\if@filesw</code>	418, 644, 683, 1213
<code>\default</code>	1104, 1139	<code>\if@safe@actives</code>	526
<code>\define@key</code>	810, 813, 816, 819, 1157, 1235, 1238, 1241	<code>\ifcase</code>	84
<code>\detokenize</code>	797	<code>\ifcsname</code>	520, 707, 730
<code>\dftest</code>	135, 142, 143, 144, 145, 146, 147, 148, 149, 150	<code>\ifeof</code>	967, 999, 1022
<code>\dimen@</code>	1255, 1260, 1272	<code>\ifetex@unexpanded</code>	217, 319
<code>\dimexpr</code>	121, 123, 1260	<code>\ifnum</code>	1197, 1259
<code>\do</code>	270, 289, 393, 428	<code>\ifodd</code>	93
<code>\documentclass</code>	3, 38, 222, 1278, 1303	<code>\ifpdf</code>	1192
<code>\dotfill</code>	137, 141	<code>\ifx</code>	196, 249, 272, 333, 429, 454, 493, 580, 750, 917, 989, 1008, 1012, 1016, 1033, 1057, 1082, 1085, 1111, 1120, 1127, 1136, 1195
<b>E</b>		<code>\ifZREF@found</code>	193, 277, 1102, 1109
<code>\emph</code>	118	<code>\ifZREF@immediate</code>	408, 420, 424, 430
<code>\end</code>	35, 98, 127, 151, 153, 1298, 1311, 1316, 1322, 1323	<code>\ifzref@titleref@expand</code>	775, 791
<code>\endcsname</code>	196, 199, 200, 239, 249, 263, 288, 301, 314, 333, 362, 363, 367, 370, 373, 383, 430, 432, 437, 438, 454, 474, 485, 486, 488, 507, 508, 509, 520, 533, 707, 708, 724, 726, 729, 730, 731, 742, 746, 750, 751, 753, 755, 756, 761, 811, 814, 943, 1042, 1055, 1064, 1149, 1195	<code>\ifzref@titleref@stripperiod</code>	788, 799
<code>\endinginput</code>	160, 213, 227, 578, 605, 623, 641, 663, 696, 769, 934, 1169, 1180, 1188, 1202, 1231	<code>\ifZREF@xr@zreflabel</code>	940, 974, 1007
<code>\etex@unexpanded</code>	328, 514, 1018	<code>\immediate</code>	413, 684
<code>\Expect</code>	1286, 1289, 1292, 1294, 1296, 1297, 1310, 1315, 1321	<code>\IncludeTests</code>	1281, 1306
<code>\ExpectRuns</code>	1315, 1321	<code>\item</code>	77, 80, 82, 90, 94, 96
<b>F</b>		<b>K</b>	
<code>\filename@area</code>	1019	<code>\kern</code>	1269
<code>\filename@parse</code>	957		
<code>\foo</code>	19, 30, 32, 34	<b>L</b>	
<code>\frontmatter</code>	73	<code>\l</code>	391
		<code>\label</code>	580, 831
		<code>\leavevmode</code>	1249
		<code>\LogTests</code>	1282, 1307
		<code>\lst@caption</code>	920
		<code>\lst@label</code>	917
		<code>\lst@MakeCaption</code>	916
		<code>\LT@c@option</code>	908
		<b>M</b>	
		<code>\m@ne</code>	646
		<code>\mainmatter</code>	102
		<code>\makeatletter</code>	10, 44, 71, 944, 1284
		<code>\makeatother</code>	17, 69
		<code>\makebox</code>	137, 138
		<code>\MessageBreak</code>	220, 969, 1035, 1076, 1184, 1199
		<b>N</b>	
		<code>\NeedsTeXFormat</code>	2, 156, 182, 574, 601, 619, 636, 659, 674, 692, 765, 930, 1165, 1176, 1227, 1276, 1302



<code>\newcommand</code> . . . . .	19, 48, 55, 61, 135, 579, 586, 595, 598, 666, 712, 760, 822, 825, 941, 1160, 1219, 1222, 1247, 1248
<code>\newcounter</code> . . . . .	5, 549, 609, 703
<code>\newif</code> . . . . .	193, 408, 775, 788, 940
<code>\newlabel</code> . . . . .	1049, 1059, 1072
<code>\newpage</code> . . . . .	111
<code>\nfss@text</code> . . . . .	560
<code>\nofiles</code> . . . . .	1277
<code>\number</code> . . . . .	64, 79, 552, 679, 698, 718, 742, 747, 1252, 1315, 1321
<code>\numexpr</code> . . . . .	64, 79, 84, 679, 718, 1029, 1038, 1051, 1079, 1259
<b>O</b>	
<code>\on@line</code> . . . . .	188, 681, 981
<code>\openin</code> . . . . .	966
<b>P</b>	
<code>\PackageError</code> . . . . .	189, 243, 257, 341
<code>\PackageInfo</code> . . . . .	240, 356, 682, 973, 982
<code>\PackageWarning</code> . . . . .	297, 310, 395, 968
<code>\PackageWarningNoLine</code> . . . . .	1034, 1075
<code>\page</code> . . . . .	1107
<code>\pdflastxpos</code> . . . . .	1207
<code>\pdflastypos</code> . . . . .	1208
<code>\pdfsavepos</code> . . . . .	1184, 1199, 1214, 1253, 1266
<code>\pdftexversion</code> . . . . .	1197
<code>\ProcessOptions</code> . . . . .	179
<code>\protect</code> . . . . .	465
<code>\protected@write</code> . . . . .	423
<code>\providecommand</code> . . . . .	. . . 231, 677, 937, 1244, 1245, 1246
<code>\ProvidesPackage</code> . . . . .	157, 183, 575, 602, 620, 637, 660, 675, 693, 766, 931, 1166, 1177, 1228
<b>R</b>	
<code>\read</code> . . . . .	997
<code>\refstepcounter</code> . . . . .	628
<code>\renewcommand</code> . . . . .	6, 551
<code>\RequirePackage</code> . . . . .	159, 164, 216, 221, 229, 577, 604, 606, 622, 639, 640, 662, 664, 665, 695, 697, 768, 770, 771, 933, 935, 1168, 1179, 1191, 1230, 1232, 1234
<code>\reset@font</code> . . . . .	560
<b>S</b>	
<code>\section</code> . . . . .	105, 113
<code>\setcounter</code> . . . . .	611
<code>\setkeys</code> . . . . .	823, 1161, 1247
<code>\setlength</code> . . . . .	1255
<code>\space</code> . . . . .	467, 984, 985, 1184, 1199
<code>\stepcounter</code> . . . . .	20, 613, 705, 706
<b>T</b>	
<code>\tableofcontents</code> . . . . .	100
<code>\the</code> . . . . .	12, 121, 123, 400, 405, 436, 450, 615, 652, 785, 1019, 1029, 1038, 1051, 1079, 1207, 1208
<code>\thechapter</code> . . . . .	13
<code>\thefoo</code> . . . . .	6, 11, 21
<code>\thetype</code> . . . . .	1064
<code>\thepage</code> . . . . .	421, 425, 467, 568, 741
<code>\thezpage</code> . . . . .	11, 741, 745
<code>\thezref@unique</code> . . . . .	8, 551, 738, 739, 744, 745, 747, 1254, 1256, 1259, 1267
<code>\title</code> . . . . .	1116, 1140
<code>\toks@</code> . . . . .	392, 399, 400, 405, 434, 436, 449, 450, 647, 652, 779, 785, 1006, 1019
<code>\TR@TitleReference</code> . . . . .	1101, 1135
<code>\ttl@sect@i</code> . . . . .	900
<b>U</b>	
<code>\unexpanded</code> . . . . .	220, 225
<code>\url</code> . . . . .	1132, 1151
<code>\usepackage</code> . . . . .	. . . 8, 40, 42, 1279, 1280, 1304, 1305
<b>V</b>	
<code>\value</code> . . . . .	12
<code>\verb</code> . . . . .	141
<b>W</b>	
<code>\write</code> . . . . .	412, 413, 684
<b>X</b>	
<code>\x</code> . . . . .	269, 271, 272, 393, 394, 396, 400, 534, 536, 717, 720, 1005, 1008, 1012, 1016, 1030, 1032, 1040, 1042, 1052, 1056
<code>\XR@ext</code> . . . . .	938
<b>Y</b>	
<code>\y</code> . . . . .	268, 272
<b>Z</b>	
<code>\z@</code> . . . . .	713, 1269
<code>\Z@L@alist</code> . . . . .	1289, 1294, 1297
<code>\Z@L@LastPage</code> . . . . .	649
<code>\Z@L@main</code> . . . . .	648, 1286, 1292, 1296
<code>\zdotfill</code> . . . . .	12, 138, 141, 1248
<code>\zdotfillsetup</code> . . . . .	13, 1247
<code>\zexternaldocument</code> . . . . .	13, 941
<code>\zlabel</code> . . . . .	9, 53, 74, 106, 114, 579
<code>\zmakeperpage</code> . . . . .	11, 712
<code>\zpageref</code> . . . . .	9, 95, 595
<code>\zposx</code> . . . . .	12, 121, 1219, 1259
<code>\zposy</code> . . . . .	12, 123, 1219
<code>\zref</code> . . . . .	9, 26, 27, 28, 29, 81, 83, 92, 97, 107, 586, 596
<code>\ZREF@C@newprop</code> . . . . .	363, 366
<code>\ZREF@C@makeperpage</code> . . . . .	713, 718, 722
<code>\ZREF@C@newprop</code> . . . . .	358, 360
<code>\ZREF@C@perpage@step</code> . . . . .	727, 735
<code>\zref@addprop</code> . . . . .	5, 14, 15, 16, 46, 293, 569, 570, 616, 625, 700, 701, 702, 774, 1173, 1209, 1210
<code>\ZREF@addtoks</code> . . . . .	448
<code>\ZREF@baseok</code> . . . . .	571
<code>\zref@default</code> . . . . .	7, 358, 557, 559
<code>\ZREF@df@dot</code> . . . . .	1242, 1246, 1272
<code>\ZREF@df@min</code> . . . . .	1239, 1244, 1260
<code>\ZREF@df@unit</code> . . . . .	1236, 1245, 1255
<code>\ZREF@dotfill</code> . . . . .	1257, 1263, 1271

<code>\ZREF@ErrorNoLine</code> .....	<code>\ZREF@org@LT@c@option</code> .....
..... 186, 206, 219, 1183, 1198	909
<code>\ZREF@extract</code> .....	<code>\ZREF@org@refstepcounter</code> .....
472, 477	630
<code>\zref@extract</code> .. 6, 65, 66, 79, 108,	<code>\ZREF@org@stepcounter</code> .....
471, 593, 744, 745, 832, 1220, 1223	705, 710
<code>\zref@extractdefault</code> .....	<code>\ZREF@org@thepage</code> .....
.... 6, 85, 86, 478, 501, 667, 747	421, 425
<code>\ZREF@foundfalse</code> .....	<code>\ZREF@org@ttl@sect@i</code> .....
267, 1100	902
<code>\ZREF@foundtrue</code> .....	<code>\ZREF@org@write</code> .....
273, 1142	412, 413
<code>\zref@getcurrent</code> .....	<code>\ZREF@P</code> .....
5, 372	357, 361, 362, 363,
<code>\ZREF@gtemp</code> .....	364, 367, 428, 430, 432, 437, 438
321, 322, 323	<code>\ZREF@patch</code> .....
<code>\ZREF@iflistcontainsprop</code> ... 263, 265	194, 627,
<code>\zref@iflistcontainsprop</code> .....	836, 842, 848, 854, 860, 866,
..... 5, 262, 296, 309	872, 879, 885, 891, 899, 907, 915
<code>\zref@iflistundefined</code> 5, 237, 248, 256	<code>\zref@prop</code> .....
<code>\zref@ifpropundefined</code> 6, 332, 340,	285, 290
394, 1060, 1094, 1113, 1122, 1129	<code>\zref@propexists</code> 6, 295, 308, 339, 587
<code>\ZREF@ifrefcontainsprop</code> ... 484, 492	<code>\ZREF@refused</code> .....
<code>\zref@ifrefcontainsprop</code> .....	461, 463
..... 7, 480, 1146, 1147	<code>\zref@refused</code> 7, 460, 592, 598, 670, 830
<code>\zref@ifrefundefined</code> .....	<code>\zref@require@unique</code> 8, 545, 704, 1233
.. 7, 453, 464, 481, 502, 739, 1256	<code>\zref@setcurrent</code> . 5, 51, 364, 369, 629
<code>\ZREF@immediatetrue</code> .....	<code>\zref@setdefault</code> .....
411	7, 556, 559
<code>\ZREF@l@addto@macro</code> .....	<code>\zref@setmainlist</code> .....
314, 319	7, 562
<code>\ZREF@label</code> .....	<code>\ZREF@stripperperiod</code> .....
381, 405, 417, 652	801, 809
<code>\zref@label</code> .....	<code>\ZREF@temp</code> .... 161, 168, 169, 170,
6, 375, 583	171, 172, 173, 174, 175, 176,
<code>\zref@labelbylist</code> 6, 376, 378, 738, 1215	177, 178, 427, 434, 435, 443, 1252
<code>\zref@labelbyprops</code> .....	<code>\ZREF@titleref</code> .....
..... 6, 58, 388, 1254, 1267	826, 828
<code>\zref@listexists</code> 5, 255, 294, 307, 380	<code>\zref@titleref@cleanup</code> .... 777, 817
<code>\zref@listforloop</code> .....	<code>\zref@titleref@current</code> .....
283	..... 772, 773, 796, 800, 801, 820
<code>\zref@localaddprop</code> . 5, 306, 1291, 1293	<code>\ZREF@titleref@hook</code> 776, 780, 784, 807
<code>\ZREF@mainlist</code> .....	<code>\zref@titleref@setcurrent</code> .. 790,
376, 563,	838, 844, 850, 856, 862, 868,
566, 569, 570, 616, 625, 774, 1173	874, 881, 887, 893, 901, 910, 919
<code>\ZREF@makeperpage@opt</code> .....	<code>\zref@titleref@stripperperiodtrue</code> . 789
713, 715	<code>\ZREF@unexpanded</code> 503, 505, 514, 516, 518
<code>\ZREF@name</code> .....	<code>\ZREF@UpdatePdfTeX</code> ... 192, 1186, 1201
185,	<code>\ZREF@wrapper@babel</code> .....
189, 243, 257, 297, 310, 341, 395	536, 542
<code>\zref@newlabel</code> .....	<code>\zref@wrapper@babel</code> .....
..... 6, 231, 233, 443, 1027, 1071	... 7, 108, 461, 519, 583, 588, 826
<code>\zref@newlist</code> .....	<code>\zref@wrapper@immediate</code> 8, 57, 409, 651
. 5, 236, 566, 642, 699, 1206, 1288	<code>\zref@wrapper@unexpanded</code> .... 8, 513
<code>\ZREF@newprop</code> .....	<code>\ZREF@X</code> .....
349, 352, 355	348, 351, 362
<code>\zref@newprop</code> .....	<code>\ZREF@xr@@input</code> .....
5, 11, 12,	1016, 1073
13, 45, 346, 567, 568, 615, 624,	<code>\ZREF@xr@checkfile</code> .... 962, 965, 993
698, 773, 936, 1061, 1095, 1114,	<code>\ZREF@xr@checkkey</code> .....
1123, 1130, 1170, 1207, 1208, 1287	1083, 1093
<code>\ZREF@nil</code> 367, 494, 509, 998, 1004,	<code>\ZREF@xr@checklist</code> .....
1009, 1013, 1027, 1040, 1049,	1040, 1081
1056, 1081, 1088, 1098, 1101, 1135	<code>\zref@xr@ext</code> .....
<code>\ZREF@NOVALUE</code> .....	13, 937, 958, 1158
500	<code>\ZREF@xr@externaldocument</code> .....
<code>\ZREF@novalue</code> .....	..... 947, 950, 953
493, 494, 500	<code>\ZREF@xr@file</code> .....
<code>\ZREF@org@@caption</code> .....	956,
839	966, 969, 974, 983, 991, 1036, 1077
<code>\ZREF@org@@chapter</code> .....	<code>\ZREF@xr@filelist</code> .....
851	... 955, 989, 991, 992, 1017, 1018
<code>\ZREF@org@@part</code> .....	<code>\ZREF@xr@found</code> .. 976, 984, 1029, 1051
845	<code>\ZREF@xr@graburl</code> .....
<code>\ZREF@org@@schapter</code> .....	958, 960
869	<code>\ZREF@xr@ignored</code> 977, 985, 1038, 1079
<code>\ZREF@org@@sect</code> .....	<code>\ZREF@xr@ignorewarning</code> .....
857	..... 1046, 1068, 1074
<code>\ZREF@org@@spart</code> .....	<code>\ZREF@xr@line</code> ... 997, 998, 1009, 1013
863	<code>\ZREF@xr@list</code> .....
<code>\ZREF@org@@ssect</code> .....	1032, 1033
875	<code>\ZREF@xr@newlabel</code> .....
<code>\ZREF@org@beamer@section</code> .....	1012, 1072
882	<code>\ZREF@xr@prefix</code> .....
<code>\ZREF@org@beamer@subsection</code> .... 888	954,
<code>\ZREF@org@beamer@subsubsection</code> . 894	1028, 1044, 1046, 1050, 1066, 1068
<code>\ZREF@org@lst@MakeCaption</code> .....	<code>\ZREF@xr@procesfile</code> .....
923	1001

\ZREF@xr@process@label ..	1013, 1049	\ZREF@xr@zreflabelfalse .....	946
\ZREF@xr@process@zreflabel	1009, 1027	\ZREF@xr@zreflabeltrue .....	949
\ZREF@xr@processfile ..	965, 996, 1024	\ZREF@zref .....	588, 591
\ZREF@xr@processline .....	998, 1004	\zrefused ..	9, 62, 63, 129, 130, 131, 598
\ZREF@xr@refname .....	1028,	\zruns .....	10, 677, 1310, 1315, 1321
	1031, 1042, 1050, 1053, 1055, 1064	\zsavepos .....	12, 125, 126, 1211
\ZREF@xr@scanparams .....	1054, 1098	\ztitleref .....	11, 825
\ZREF@xr@scantitleref ...	1101, 1135	\ztitlerefsetup .....	12, 810
\ZREF@xr@url .....	961, 1151	\ztotpages .....	10, 93, 666
\ZREF@xr@urlcheck ..	1044, 1066, 1145	\zunmakeperpage .....	11, 760
\ZREF@xr@zref@newlabel ..	1008, 1071	\zxrsetup .....	13, 1157