

# The zref package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2010/04/17 v2.12

## 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	4
1.3	Interfaces	4
<b>2</b>	<b>Interface for programmers</b>	<b>4</b>
2.1	Entities	5
2.2	Property list	5
2.3	Property	6
2.4	Reference generation	6
2.5	Data extraction	7
2.6	Setup	7
2.7	Declared properties	8
2.8	Wrapper for advanced situations	8
2.9	Counter for unique names	8
<b>3</b>	<b>User interface</b>	<b>9</b>
3.1	Module <code>user</code>	9
3.2	Module <code>abspage</code>	10
3.3	Module <code>lastpage</code>	10
3.3.1	Tests for last page	10
3.3.2	Example	10
3.4	Module <code>thepage</code>	11
3.5	Module <code>nextpage</code>	12
3.5.1	Configuration	12
3.5.2	Example	12
3.6	Module <code>totpages</code>	13
3.7	Module <code>marks</code>	13
3.8	Module <code>runs</code>	13
3.9	Module <code>perpage</code>	13
3.10	Module <code>counter</code>	14
3.11	Module <code>titleref</code>	14
3.12	Module <code>savepos</code>	15
3.13	Module <code>dotfill</code>	15
3.14	Module <code>xr</code>	16

<b>4</b>	<b>ToDo</b>	<b>16</b>
<b>5</b>	<b>Example</b>	<b>17</b>
<b>6</b>	<b>Implementation</b>	<b>19</b>
6.1	Package <code>zref</code>	19
6.1.1	Identification	19
6.1.2	Load basic module	19
6.1.3	Process options	19
6.2	Module <code>base</code>	20
6.2.1	Prefixes	20
6.2.2	Identification	20
6.2.3	Utilities	20
6.2.4	Check for $\varepsilon$ -TeX	21
6.2.5	Auxiliary file stuff	21
6.2.6	Property lists	21
6.2.7	Properties	23
6.2.8	Reference generation	24
6.2.9	Reference querying and extracting	26
6.2.10	Compatibility with <code>babel</code>	28
6.2.11	Unique counter support	28
6.2.12	Utilities	29
6.2.13	Setup	29
6.3	Module <code>user</code>	30
6.4	Module <code>abspage</code>	30
6.5	Module <code>counter</code>	31
6.6	Module <code>lastpage</code>	31
6.7	Module <code>thepage</code>	32
6.8	Module <code>nextpage</code>	33
6.9	Module <code>totpages</code>	34
6.10	Module <code>marks</code>	35
6.11	Module <code>runs</code>	36
6.12	Module <code>perpage</code>	37
6.13	Module <code>titleref</code>	38
6.13.1	Implementation	38
6.13.2	User interface	40
6.13.3	Patches for section and caption commands	40
6.13.4	Class <code>memoir</code>	41
6.13.5	Class <code>beamer</code>	42
6.13.6	Package <code>titlesec</code>	42
6.13.7	Package <code>longtable</code>	42
6.13.8	Package <code>listings</code>	43
6.13.9	Theorems	43
6.14	Module <code>xr</code>	43
6.15	Module <code>hyperref</code>	50
6.16	Module <code>savepos</code>	50
6.16.1	Identification	50
6.16.2	Availability	50
6.16.3	Setup	51
6.16.4	User macros	51
6.17	Module <code>dotfill</code>	51
<b>7</b>	<b>Test</b>	<b>52</b>
7.1	<code>\zref@localaddprop</code>	52
7.2	Module <code>runs</code>	53
7.3	Module <code>titleref</code>	53

<b>8</b>	<b>Installation</b>	<b>54</b>
8.1	Download . . . . .	54
8.2	Bundle installation . . . . .	55
8.3	Package installation . . . . .	55
8.4	Refresh file name databases . . . . .	55
8.5	Some details for the interested . . . . .	55
<b>9</b>	<b>References</b>	<b>56</b>
<b>10</b>	<b>History</b>	<b>56</b>
	[2006/02/20 v1.0] . . . . .	56
	[2006/05/03 v1.1] . . . . .	57
	[2006/05/25 v1.2] . . . . .	57
	[2006/09/08 v1.3] . . . . .	57
	[2007/01/23 v1.4] . . . . .	57
	[2007/02/18 v1.5] . . . . .	57
	[2007/04/06 v1.6] . . . . .	57
	[2007/04/17 v1.7] . . . . .	57
	[2007/04/22 v1.8] . . . . .	57
	[2007/05/02 v1.9] . . . . .	57
	[2007/05/06 v2.0] . . . . .	57
	[2007/05/28 v2.1] . . . . .	57
	[2008/09/21 v2.2] . . . . .	58
	[2008/10/01 v2.3] . . . . .	58
	[2009/08/07 v2.4] . . . . .	58
	[2009/12/06 v2.5] . . . . .	58
	[2009/12/07 v2.6] . . . . .	58
	[2009/12/08 v2.7] . . . . .	58
	[2010/03/26 v2.8] . . . . .	58
	[2010/03/29 v2.9] . . . . .	58
	[2010/04/08 v2.10] . . . . .	58
	[2010/04/15 v2.11] . . . . .	58
	[2010/04/17 v2.12] . . . . .	59
<b>11</b>	<b>Index</b>	<b>59</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 is fixed in the  $\text{\LaTeX}$  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  $\text{\LaTeX}$  and other packages.

**titleref** stores its title data into the first entry in the list.  $\text{\LaTeX}$  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_{your key}_key{some text}
```

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  $\text{\LaTeX}3$  that will need some time before its first release. Thus I have implemented it as  $\text{\LaTeX}2_{\epsilon}$  package without disturbing the existing  $\text{\LaTeX}$  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 package 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:

$$\backslash\mathrm{Z@R@myref} \rightarrow \backslash\mathrm{default}\{2.1\}\backslash\mathrm{page}\{7\}$$

The generic format of a entry is:

$$\backslash\mathrm{Z@R@}\langle\mathrm{refname}\rangle \rightarrow \backslash\langle\mathrm{propname}\rangle\{\langle\mathrm{value}\rangle\}$$

$\langle\mathrm{refname}\rangle$  is the name that denoted references (the name used in  $\backslash\mathrm{label}$  and  $\backslash\mathrm{ref}$ ).  $\langle\mathrm{propname}\rangle$  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  $\backslash\mathrm{label}$  command is the *main property list*.

## 2.2 Property list

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

$\backslash\mathrm{zref@newlist} \{\langle\mathrm{listname}\rangle\}$

Declares a new empty property list.

$\backslash\mathrm{zref@addprop} \{\langle\mathrm{listname}\rangle\} \{\langle\mathrm{propname}\rangle\}$

Adds the property  $\langle\mathrm{propname}\rangle$  to the property list  $\langle\mathrm{listname}\rangle$ . The property and list must exist.

$\backslash\mathrm{zref@localaddprop} \{\langle\mathrm{listname}\rangle\} \{\langle\mathrm{propname}\rangle\}$

Local variant of  $\backslash\mathrm{zref@addprop}$ .

$\backslash\mathrm{zref@listexists} \{\langle\mathrm{listname}\rangle\} \{\langle\mathrm{then}\rangle\}$

Executes  $\langle\mathrm{then}\rangle$  if the property list  $\langle\mathrm{listname}\rangle$  exists or raise an error otherwise.

$\backslash\mathrm{zref@iflistundefined}^{\mathrm{exp}} \{\langle\mathrm{listname}\rangle\} \{\langle\mathrm{then}\rangle\} \{\langle\mathrm{else}\rangle\}$

Executes  $\langle\mathrm{then}\rangle$  if the list exists or  $\langle\mathrm{else}\rangle$  otherwise.

$\backslash\mathrm{zref@iflistcontainsprop} \{\langle\mathrm{listname}\rangle\} \{\langle\mathrm{propname}\rangle\} \{\langle\mathrm{then}\rangle\} \{\langle\mathrm{else}\rangle\}$

Executes  $\langle\mathrm{then}\rangle$  if the property  $\langle\mathrm{propname}\rangle$  is part of property list  $\langle\mathrm{listname}\rangle$  or otherwise it runs the  $\langle\mathrm{else}\rangle$  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 `<then>` if the property `<propname>` is available or generates an error message otherwise.

`\zref@ifpropundefinedexp {<propname>} {<then>} {<else>}`

Calls `<then>` or `<else>` depending on the existence of property `<propname>`.

## 2.4 Reference generation

`\zref@label {<refname>}`

This works similar to `\label`. The reference `<refname>` 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 `<listname>`.

`\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 *<propname>* for the reference *<refname>*. In case of errors such as undefined reference the *<default>* 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:

```

 $\text{\LaTeX:}$   \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.

`\zifrefundefined {<refname>} {<then>} {<else>}`

Macro `\zifrefundefined` calls `\zref@refused` before executing `\zref@ifrefundefined`. Babel shorthands are supported in *<refname>*.

`\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 {\langle value \rangle}`

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>	<code>0</code>
<code>perpage</code>	<code>pagevalue</code>	<code>perpage</code>	<code>0</code>
	<code>page</code>	<code>perpage</code>	<code>&lt;empty&gt;</code>
	<code>abspage</code>	<code>perpage</code>	<code>0</code>
<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>	<code>0</code>
	<code>posy</code>	<code>savepos</code>	<code>0</code>
<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 {...} {\langle name \rangle}`

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:

<code>babel</code>	Babel shorthands are allowed.
<code>robust</code>	Robust macro.
<code>exp</code>	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`:

$$\text{\zref}\{x\} \equiv \text{\zref}[\text{default}]\{x\}$$

`\zpageref {⟨refname⟩}^babel`

Convenience macro, similar to `\pageref`.

$$\text{\zpageref}\{x\} \equiv \text{\zref}[\text{page}]\{x\}$$

`\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:

```
\zref@extract{LastPage}{page} (+ \zref@refused{LastPage})
```

or

```
\zpageref{LastPage} (module user)
```

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 Tests for last page

Since version 2010/03/26 v2.8 the macros `\zref@iflastpage` and `\ziflastpage` were added. They test the reference, whether it is a reference of the last page.

`\zref@iflastpageexp {<refname>} {<then>} {<else>}`

Macro `\zref@iflastpage` compares the references `<refname>` with `<LastPage>`. Basis of the comparison is the value of property `abspage`, because the values are different for different pages. This is not ensured by property `page`. Therefore module `abspage` is loaded by module `lastpage`. If both values of property `abspage` are present and match, then `<then>` is executed, otherwise code `<else>` is called. If one or both references are undefined or lack the property `abspage`, then `<else>` is executed.

Macro `\zref@iflastpage` is expandable, therefore `\zref@refused` should be called on `<refname>` and `<LastPage>`.

`\ziflastpage {<refname>} {<then>} {<else>}`

Macro `\ziflastpage` has the same function as `\zref@iflastpage`, but adds support for babel shorthands in `<refname>` and calls `\zref@refused`. However macro `\ziflastpage` is not expandable.

#### 3.3.2 Example

```
1 <*example-lastpage>
2 %<<END_EXAMPLE
3 \NeedsTeXFormat{LaTeX2e}
4 \documentclass{report}
5
6 \newcounter{foo}
7 \renewcommand*{\thefoo}{\Alph{foo}}
```

```

8
9 \usepackage{zref-lastpage,zref-user}[2010/04/17]
10
11 \makeatletter
12 \zref@newprop{thefoo}{\thefoo}
13 \zref@newprop{valuefoo}{\the\value{foo}}
14 \zref@newprop{chapter}{\thechapter}
15 \zref@addprop{LastPage}{thefoo}
16 \zref@addprop{LastPage}{valuefoo}
17 \zref@addprop{LastPage}{chapter}
18 \makeatother
19
20 \newcommand*{\foo}{%
21   \stepcounter{foo}%
22   [Current foo: \thefoo]%
23 }
24
25 \begin{document}
26   \chapter{First chapter}
27   Last page is \zref{LastPage}.\
28   Last chapter is \zref[chapter]{LastPage}.\
29   Last foo is \zref[thefoo]{LastPage}.\
30   Last value of foo is \zref[valuefoo]{LastPage}.\
31   \foo
32   \chapter{Second chapter}
33   \foo\foo\foo
34   \chapter{Last chapter}
35   \foo
36 \end{document}
37 %END_EXAMPLE
38 \</example-lastpage>

```

### 3.4 Module thepage

This module `thepage` loads module `abspage`, constructs a reference name using the absolute page number and remembers property `page`. Other properties can be added by adding them to the property list `thepage`.

`\zthepage {⟨absolute page number⟩}`

Macro `\zthepage` is basically a `\zpageref`. The reference name is yield by the `⟨absolute page number⟩`. If the reference is not defined, then the default for property `page` is used.

`\zref@thepage@nameexp {⟨absolute page number⟩}`

Macro `\zref@thepage@name` returns the internal reference name that is constructed using the `⟨absolute page number⟩`. The internal reference name should not be used directly, because it might change in future versions.

`\zref@thepageexp {⟨absolute page number⟩}`  
`\zref@thepage@refused {⟨absolute page number⟩}`

Macro `\zref@thepage` returns the page number (`\thepage`) of `⟨absolute page number⟩`. Because this macro is expandable, `\zref@thepage@refused` is used outside an expandable context to mark the reference as used.

## 3.5 Module `nextpage`

`\znextpage`

Macro `\znextpage` prints `\thepage` of the following page. It gets the current absolute page number by using a label. There are three cases for the next page:

1. The next page is not known yet because of undefined references. Then `\zunknownnextpagename` is used instead. The default for this macro is the default of property `page`.
2. This page is the last page. Then `\znonextpagename` is used. Its default is empty.
3. The next page is known, then `\thepage` of the next page is used (the value of property `page` of the next page).

### 3.5.1 Configuration

The behaviour can be configured by the following macros.

`\zunknownnextpagename`  
`\znonextpagename`

If the next page is not known or available, then `\znextpage` uses these name macros as default. `\zunknownnextpagename` is used in case of undefined references. Default is the value of property `page` of the next page (`\thepage`). Module `thepage` is used.

Macro `\znonextpagename` is used, if the next page does not exists. That means that the current page is last page. The default is empty.

`\znextpagesetup {<unknown>} {<no next>} {<next>}`

According to the case (see `\znextpage`) macro `\znextpage` calls an internal macro with an argument. The argument is either `\thepage` of the next page or one of `\zunknownnextpagename` or `\znonextpagename`. These internal macro can be changed by `\znextpagesetup`. It expects the definition texts for these three cases of a macro with one argument. The default is

`\znextpagesetup{#1}{#1}{#1}`

### 3.5.2 Example

```
39 <*example-nextpage>
40 %<<END_EXAMPLE
41 \documentclass{book}
42
43 \usepackage{zref-nextpage}[2010/04/17]
44 \znextpagesetup
45   {\thepage}% next page is unknown
46   {\thepage\ (#1)}% this page is last page
47   {\thepage\ $\rightarrow$ #1}% next page is known
48 \renewcommand*{\znonextpagename}{last page}
49
50 \usepackage{fancyhdr}
51 \pagestyle{fancy}
52 \fancyhf{}
53 \fancyhead[LE,RO]{\znextpage}
54 \fancypagestyle{plain}{%
55   \fancyhf{}}
```

```

56 \fancyhead[LE,R0]{\znexthpage}%
57 }
58
59 \begin{document}
60 \frontmatter
61 \tableofcontents
62 \mainmatter
63 \chapter{Hello World}
64 \clearpage
65 \section{Last section}
66 \end{document}
67 %END_EXAMPLE
68 \end{example-nextpage}

```

### 3.6 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.

### 3.7 Module `marks`

ToDo.

### 3.8 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`).

### 3.9 Module `perpage`

With `\@addtoreset` or `\numberwithin` a counter can be reset if another counter is incremented. This does 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  $\text{\LaTeX}$  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.

### 3.10 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.

### 3.11 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.

### 3.12 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  $\langle refname \rangle$  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.

### 3.13 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`).

### 3.14 Module `xr`

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

`\xexternaldocument * [<prefix>] babel {<external document>} [<url>]`

See `\xexternaldocument` for a description of this option. The found labels also get a property `externaldocument` that remembers *<external document>*. 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, ...}`

The following setup options are available:

**ext:** It sets the default extension.

**tozreflabel:** The found references are imported as zref labels. This is enabled by default.

**toltxlabel:** The found references are imported as L<sup>A</sup>T<sub>E</sub>X labels. Packages `nameref`, `titleref` and class `memoir` are supported.

**verbose:** List the imported labels in the `.log` file. Default is `false`.

`\zref@xr@ext`

If the *<url>* 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

```
69 \example
70 \documentclass{book}
71
72 \usepackage[ngerman]{babel}%
73
74 \usepackage[savepos,totpages,titleref,dotfill,counter,user]{zref}
75
```

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.

```
76 \makeatletter
77 \zref@newprop{chaptitle}{}
78 \zref@addprop{main}{chaptitle}
79
80 \newcommand*{\ChapterStart}[2]{%
81   \cleardoublepage
82   \def\current@chapid{#1}%
83   \zref@setcurrent{chaptitle}{#2}%
84   \chapter{#2}%
85   \zlabel{chap:#1}%
86 }
87 \newcommand*{\ChapterStop}{%
88   \cleardoublepage
89   \zref@wrapper@immediate{%
90     \zref@labelbyprops{chapend:\current@chapid}{abspage}%
91   }%
92 }
```

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

```
93 \newcommand*{\ChapterPages}[1]{%
94   \zrefused{chap:#1}%
95   \zrefused{chapend:#1}%
96   \number\numexpr
97     \zref@extract{chapend:#1}{abspage}%
98     -\zref@extract{chap:#1}{abspage}%
99   +1\relax
100 }
101 \makeatother
102 \begin{document}
```

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

```
103 \makeatletter
104
105 \frontmatter
106 \zlabel{documentstart}
107
108 \begin{itemize}
109 \item
110   The frontmatter part has
111   \number\numexpr\zref@extract{chap:first}{abspage}-1\relax~pages.
112 \item
113   Chapter \zref{chap:first} has \ChapterPages{first} page(s).
114 \item
```

```

115 Section \zref{hello} is on the
116 \ifcase\numexpr
117   \zref@extractdefault{hello}{page}{0}%
118   -\zref@extractdefault{chap:first}{page}{0}%
119   +1\relax
120   ??\or first\or second\or third\or forth\fi
121   ~page inside its chapter.
122 \item
123 The document has
124 \zref[abspage]{LastPage} pages.
125 This number is \ifodd\ztotpages odd\else even\fi.
126 \item
127 The last page is labeled with \zpageref{LastPage}.
128 \item
129 The title of chapter \zref{chap:next} is ‘‘\zref[chaptitle]{chap:next}’’.
130 \end{itemize}
131
132 \tableofcontents
133
134 \mainmatter
135 \ChapterStart{first}{First chapter}
136

```

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`.

```

137 \section{Test}
138 \zlabel{a"o}
139 Section \zref{a"o} on page
140 \zref@wrapper@babel\zref@extract{a"o}{page}.
141
142 Text.
143 \newpage
144
145 \section{Hello World}
146 \zlabel{hello}
147
148 \ChapterStop
149
150 \ChapterStart{next}{Next chapter with \emph{umlauts}: "a"o"u"s}
151

```

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.

```

152 The width of the first column is
153 \the\dimexpr \zposx{secondcol}sp - \zposx{firstcol}sp\relax,\,
154 the height difference of the two baselines is
155 \the\dimexpr \zposy{firstcol}sp - \zposy{secondline}sp\relax:\,
156 \begin{tabular}{ll}
157   \zsavepos{firstcol}Hello&\zsavepos{secondcol}World\,
158   \zsavepos{secondline}Second line&\foobar\,
159 \end{tabular}
160

```

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.

```

161 \zrefused{firstcol}
162 \zrefused{secondcol}
163 \zrefused{secondline}
164
165 \ChapterStop

```

Test for module `\dotfill`.

```

166 \ChapterStart{dotfill}{Test for dotfill feature}
167 \newcommand*{\dftest}[1]{%
168   #1&
169   [\makebox[#1]{\dotfill}]&
170   [\makebox[#1]{\zdotfill}]\
171 }
172 \begin{tabular}{rll}
173 & [\verb|\dotfill|] & [\verb|\zdotfill|]\
174 \dftest{0.43em}
175 \dftest{0.44em}
176 \dftest{0.45em}
177 \dftest{0.87em}
178 \dftest{0.88em}
179 \dftest{0.89em}
180 \dftest{1.31em}
181 \dftest{1.32em}
182 \dftest{1.33em}
183 \end{tabular}
184 \ChapterStop
185 \end{document}
186 \example

```

## 6 Implementation

### 6.1 Package zref

#### 6.1.1 Identification

```

187 \package
188 \NeedsTeXFormat{LaTeX2e}
189 \ProvidesPackage{zref}
190 [2010/04/17 v2.12 New reference scheme for LaTeX2e (HO)]%

```

#### 6.1.2 Load basic module

```

191 \RequirePackage{zref-base}[2010/04/17]

```

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

```

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

```

#### 6.1.3 Process options

Known modules are loaded and the release date is checked.

```

193 \def\ZREF@temp#1{%
194   \DeclareOption{#1}{%
195     \AtEndOfPackage{%
196       \RequirePackage{zref-#1}[2010/04/17]%
197     }%
198   }%
199 }
200 \ZREF@temp{abspage}
201 \ZREF@temp{counter}
202 \ZREF@temp{dotfill}
203 \ZREF@temp{hyperref}
204 \ZREF@temp{lastpage}
205 \ZREF@temp{perpage}
206 \ZREF@temp{savepos}
207 \ZREF@temp{titleref}
208 \ZREF@temp{totpages}
209 \ZREF@temp{user}
210 \ZREF@temp{xr}
211 \ProcessOptions\relax
212 \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

```
213 <*base>
214 \NeedsTeXFormat{LaTeX2e}
215 \ProvidesPackage{zref-base}%
216 [2010/04/17 v2.12 Module base for zref (H0)]%
```

### 6.2.3 Utilities

```
217 \RequirePackage{ltxcmds}[2010/03/01]
```

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

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

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

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

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

```
220 \newif\ifZREF@found
```

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

```
221 \def\ZREF@patch#1{%
222   \begingroup\expandafter\expandafter\expandafter\endgroup
223   \expandafter\ifx\csname #1\endcsname\relax
224     \expandafter\ltx@gobble
225   \else
226     \expandafter\let\csname ZREF@org@#1\endcsname
227     \csname #1\endcsname
228     \expandafter\ltx@firstofone
229   \fi
230 }
```

## 6.2.4 Check for $\epsilon$ -TeX

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

```

231 \ltX@ifundefined{eTeXversion}{%
232   \PackageError{ZREF@name}{%
233     Missing support for eTeX; package is abandoned%
234   }{%
235     Use a TeX compiler that support eTeX and enable eTeX %
236     in the format.%
237   }%
238 \endinput
239 }{}%

240 \RequirePackage{etexcmds}[2007/09/09]
241 \ifetex@unexpanded
242 \else
243   \PackageError{ZREF@name}{%
244     Missing e-TeX's \string\unexpanded.\MessageBreak
245     Add \string\RequirePackage\string{etexcmds\string} before %
246     \string\documentclass%
247   }{%
248     Probably you are using some package (e.g. ConTeXt) that %
249     redefines \string\unexpanded%
250   }%
251 \expandafter\endinput
252 \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.

```

253 \RequirePackage{auxhook}
254 \AddLineBeginAux{%
255   \string\providecommand\string\zref@newlabel[2]{}%
256 }

```

`\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.

```

257 \def\zref@newlabel{%
258   \@newl@bel{Z@R}%
259 }

```

## 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.

```

260 \def\zref@newlist#1{%
261   \zref@iflistundefined{#1}{%
262     \@ifdefinable{Z@L@#1}{%
263       \global\expandafter\let\csname Z@L@#1\endcsname\@empty
264       \PackageInfo{zref}{New property list: #1}%
265     }%
266   }{%

```

```

267     \PackageError\ZREF@name{%
268         Property list '#1' already exists%
269     }\@ehc
270 }%
271 }

```

`\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.

```

272 \def\zref@iflistundefined#1{%
273     \expandafter\ifx\csname Z@L@#1\endcsname\relax
274     \expandafter\ltx@firstoftwo
275     \else
276         \expandafter\ltx@secondoftwo
277     \fi
278 }

```

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

```

279 \def\zref@listexists#1{%
280     \zref@iflistundefined{#1}{%
281         \PackageError\ZREF@name{%
282             Property list '#1' does not exist%
283         }\@ehc
284     }%
285 }

```

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

```

286 \def\zref@iflistcontainsprop#1{%
287     \expandafter\ZREF@iflistcontainsprop\csname Z@L@#1\endcsname
288 }
289 \def\ZREF@iflistcontainsprop#1#2{%
290     \begingroup
291     \ZREF@foundfalse
292     \edef\y{#2}%
293     \expandafter\@tfor\expandafter\x
294     \expandafter:\expandafter=#1\do{%
295         \edef\x{x}%
296         \ifx\x\y
297             \ZREF@foundtrue
298         \fi
299     }%
300     \expandafter\endgroup
301     \ifZREF@found
302         \expandafter\ltx@firstoftwo
303     \else
304         \expandafter\ltx@secondoftwo
305     \fi
306 }

```

`\zref@listforloop`

```

307 \def\zref@listforloop#1#2{%
308     \expandafter\expandafter\expandafter\@tfor
309     \expandafter\expandafter\expandafter\zref@prop
310     \expandafter\expandafter\expandafter:%
311     \expandafter\expandafter\expandafter=%
312     \csname Z@L@#1\endcsname
313     \do{%
314         #2\zref@prop
315     }%
316 }

```

`\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.

```

317 \def\zref@addprop#1#2{%
318   \zref@listexists{#1}{%
319     \zref@propexists{#2}{%
320       \zref@iflistcontainsprop{#1}{#2}{%
321         \PackageWarning\ZREF@name{%
322           Property ‘#2’ is already in list ‘#1’%
323         }%
324       }{%
325         \edef\ZREF@temp{#2}%
326         \expandafter\g@addto@macro\csname Z@L@#1\expandafter\endcsname
327           \expandafter{\expandafter{\ZREF@temp}}%
328       }%
329     }%
330   }%
331 }
```

`\zref@localaddprop`

```

332 \def\zref@localaddprop#1#2{%
333   \zref@listexists{#1}{%
334     \zref@propexists{#2}{%
335       \zref@iflistcontainsprop{#1}{#2}{%
336         \PackageWarning\ZREF@name{%
337           Property ‘#2’ is already in list ‘#1’%
338         }%
339       }{%
340         \expandafter\ZREF@l@addto@macro\csname Z@L@#1\endcsname{#{#2}}%
341       }%
342     }%
343   }%
344 }
```

`\ZREF@l@addto@macro`

```

345 \ifetex@unexpanded
346   \def\ZREF@l@addto@macro#1#2{%
347     \global\let\ZREF@temp#1%
348     \g@addto@macro\ZREF@temp{#2}%
349     \let#1\ZREF@temp
350   }%
351 \else
352   \def\ZREF@l@addto@macro#1#2{%
353     \edef#1{%
354       \etex@unexpanded\expandafter{#1#2}%
355     }%
356   }%
357 \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.

```

358 \def\zref@ifpropundefined#1{%
359   \expandafter\ifx\csname Z@E@#1\endcsname\relax
360     \expandafter\ltx@firstoftwo
361   \else
362     \expandafter\ltx@secondoftwo
363   \fi
364 }
```

`\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.

```

365 \def\zref@propexists#1{%
366   \zref@ifpropundefined{#1}{%
367     \PackageError\ZREF@name{%
368       Property ‘#1’ does not exist%
369     }\@ehc
370   }%
371 }

```

`\zref@newprop` A new property is declared by `\zref@newprop`, the property name  $\langle propname \rangle$  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.

```

372 \def\zref@newprop{%
373   \@ifstar{%
374     \let\ZREF@X\noexpand
375     \ZREF@newprop
376   }{%
377     \let\ZREF@X\empty
378     \ZREF@newprop
379   }%
380 }
381 \def\ZREF@newprop#1{%
382   \PackageInfo{zref}{New property: #1}%
383   \def\ZREF@P{#1}%
384   \@ifnextchar[\ZREF@@newprop{\ZREF@@newprop[\zref@default]}%
385 }
386 \def\ZREF@@newprop[#1]{%
387   \global\@namedef{Z@D@\ZREF@P}{#1}%
388   \global\expandafter\let\csname Z@X@\ZREF@P\endcsname\ZREF@X
389   \expandafter\ZREF@@@newprop\csname\ZREF@P\endcsname
390   \expandafter\gdef\csname Z@C@\ZREF@P\endcsname{}%
391   \zref@setcurrent\ZREF@P
392 }
393 \def\ZREF@@@newprop#1{%
394   \expandafter\gdef\csname Z@E@\ZREF@P\endcsname##1#1##2##3\ZREF@nil{##2}%
395 }

```

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

```

396 \def\zref@setcurrent#1{%
397   \expandafter\def\csname Z@C@#1\endcsname
398 }

```

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

```

399 \def\zref@getcurrent#1{%
400   \csname Z@C@#1\endcsname
401 }

```

### 6.2.8 Reference generation

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

```

402 \def\zref@label#1{%
403   \zref@labelbylist{#1}\ZREF@mainlist
404 }

```



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

```

405 \def\zref@labelbylist#1#2{%
406   \@bsphack
407   \zref@listexists{#2}{%
408     \expandafter\expandafter\expandafter\ZREF@label
409     \expandafter\expandafter\expandafter{%
410       \csname Z@L@#2\endcsname
411     }{#1}%
412   }%
413   \@esphack
414 }

```

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

```

415 \def\zref@labelbyprops#1#2{%
416   \@bsphack
417   \begingroup
418   \edef\l{#2}%
419   \toks@{}%
420   \@for\x:=#2\do{%
421     \zref@ifpropundefined{\x}{%
422       \PackageWarning\ZREF@name{%
423         Property ‘\x’ is not known%
424       }%
425     }{%
426       \toks@\expandafter\expandafter\expandafter{%
427         \expandafter\the\expandafter\toks@\expandafter{\x}%
428       }%
429     }%
430   }%
431   \expandafter\endgroup
432   \expandafter\ZREF@label\expandafter{\the\toks@}{#1}%
433   \@esphack
434 }

```

`\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.

```

435 \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`.

```

436 \long\def\zref@wrapper@immediate#1{%
437   \begingroup
438   \ZREF@immediatetrue
439   \let\ZREF@org@write\write
440   \def\write{\immediate\ZREF@org@write}%
441   #1%
442   \endgroup
443 }

```

`\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.

```

444 \def\ZREF@label#1#2{%
445   \if@filesw
446     \begingroup
447     \ifZREF@immediate
448       \let\ZREF@org@thepage\thepage
449     \fi

```

```

450 \protected@write\@auxout{%
451 \ifZREF@immediate
452 \let\thepage\ZREF@org@thepage
453 \fi
454 \let\ZREF@temp\@empty
455 \@tfor\ZREF@P:=#1\do{%
456 \expandafter\ifx
457 \csname\ifZREF@immediate relax\else Z@X@\ZREF@P\fi\endcsname
458 \noexpand
459 \expandafter\let\csname Z@C@\ZREF@P\endcsname\relax
460 \fi
461 \toks@\expandafter{\ZREF@temp}%
462 \edef\ZREF@temp{%
463 \the\toks@
464 \expandafter\string\csname\ZREF@P\endcsname{%
465 \expandafter\noexpand\csname Z@C@\ZREF@P\endcsname
466 }%
467 }%
468 }%
469 }{%
470 \string\zref@newlabel{#2}{\ZREF@temp}%
471 }%
472 \endgroup
473 \fi
474 }
475 \def\ZREF@addtoks#1{%
476 \toks@\expandafter\expandafter\expandafter{%
477 \expandafter\the\expandafter\toks@#1%
478 }%
479 }

```

### 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.

```

480 \def\zref@ifrefundefined#1{%
481 \expandafter\ifx\csname Z@R@#1\endcsname\relax
482 \expandafter\ltx@firstoftwo
483 \else
484 \expandafter\ltx@secondoftwo
485 \fi
486 }

```

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

```

487 \newcommand*\zifrefundefined}[1]{%
488 \zref@wrapper@babel\ZREF@ifrefundefined{#1}%
489 }

```

`\ZREF@ifrefundefined`

```
490 \def\ZREF@ifrefundefined#1{%
491   \zref@refused{#1}%
492   \zref@ifrefundefined{#1}%
493 }
```

`\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 L<sup>A</sup>T<sub>E</sub>X and a warning is given.

```
494 \def\zref@refused#1{%
495   \zref@wrapper@babel\ZREF@refused{#1}%
496 }
```

`\ZREF@refused`

```
497 \def\ZREF@refused#1{%
498   \zref@ifrefundefined{#1}{%
499     \protect\G@refundefinedtrue
500     \@latex@warning{%
501       Reference ‘#1’ on page \thepage \space undefined%
502     }%
503   }{}%
504 }
```

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

```
505 \def\zref@extract#1#2{%
506   \expandafter\expandafter\expandafter\ZREF@extract
507   \expandafter\expandafter\expandafter{%
508     \csname Z@D@#2\endcsname
509   }{#1}{#2}%
510 }
511 \def\ZREF@extract#1#2#3{%
512   \zref@extractdefault{#2}{#3}{#1}%
513 }
```

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

```
514 \def\zref@ifrefcontainsprop#1#2{%
515   \zref@ifrefundefined{#1}{%
516     \ltx@secondoftwo
517   }{%
518     \expandafter\ZREF@ifrefcontainsprop
519     \csname Z@E@#2\expandafter\endcsname
520     \csname#2\expandafter\expandafter\expandafter\endcsname
521     \expandafter\expandafter\expandafter{%
522       \csname Z@R@#1\endcsname
523     }%
524   }%
525 }
526 \def\ZREF@ifrefcontainsprop#1#2#3{%
527   \expandafter\ifx\expandafter\ZREF@novalue
528   #1#3#2\ZREF@novalue\ZREF@nil\@empty
529   \expandafter\ltx@secondoftwo
530   \else
531   \expandafter\ltx@firstoftwo
532   \fi
533 }
534 \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.

```

535 \def\zref@extractdefault#1#2#3{%
536   \zref@ifrefundefined{#1}{%
537     \ZREF@unexpanded{#3}%
538   }{%
539     \expandafter\expandafter\expandafter\ZREF@unexpanded
540     \expandafter\expandafter\expandafter{%
541       \csname Z@E@#2\expandafter\expandafter\expandafter\endcsname
542       \csname Z@R@#1\expandafter\endcsname
543       \csname#2\endcsname{#3}\ZREF@nil
544     }%
545   }%
546 }

```

`\zref@wrapper@unexpanded`

```

547 \long\def\zref@wrapper@unexpanded#1{%
548   \let\ZREF@unexpanded\etex@unexpanded
549   #1%
550   \let\ZREF@unexpanded\ltx@firstofone
551 }
552 \let\ZREF@unexpanded\ltx@firstofone

```

## 6.2.10 Compatibility with babel

`\zref@wrapper@babel`

```

553 \long\def\zref@wrapper@babel#1#2{%
554   \ifcsname if@safe@actives\endcsname
555     \expandafter\ltx@firstofone
556   \else
557     \expandafter\ltx@secondoftwo
558   \fi
559   {%
560     \if@safe@actives
561       \expandafter\ltx@secondoftwo
562     \else
563       \expandafter\ltx@firstoftwo
564     \fi
565     {%
566       \begingroup
567         \csname @safe@activestrue\endcsname
568         \edef\x{#2}%
569       \expandafter\endgroup
570       \expandafter\ZREF@wrapper@babel\expandafter{\x}{#1}%
571     }%
572   }{%
573     #1{#2}%
574   }%
575 }
576 \long\def\ZREF@wrapper@babel#1#2{%
577   #2{#1}%
578 }

```

## 6.2.11 Unique counter support

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

```

579 \def\zref@require@unique{%
580   \@ifundefined{c@zref@unique}{%
581     \begingroup
582       \let\@addtoreset\ltx@gobbletwo
583       \newcounter{zref@unique}%
584     \endgroup

```

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

```

585 \renewcommand*{\thezref@unique}{%
586   zref@number\c@zref@unique
587 }%
588 }{}%
589 }

```

### 6.2.12 Utilities

`\ZREF@number`

```

590 \ltx@ifundefined{numexpr}{%
591   \let\ZREF@number\number
592 }{%
593   \def\ZREF@number#1{\the\numexpr#1}%
594 }

```

### 6.2.13 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`.

```

595 \def\zref@setdefault#1{%
596   \def\zref@default{#1}%
597 }

```

`\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.

```

598 \zref@setdefault{%
599   \nfss@text{\reset@font\bfseries ??}%
600 }

```

#### Main property list.

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

```

601 \def\zref@setmainlist#1{%
602   \def\ZREF@mainlist{#1}%
603 }
604 \zref@setmainlist{main}

```

Now we create the list.

```

605 \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

```

606 \zref@newprop{default}{\@currentlabel}
607 \zref@newprop*{page}{\thepage}
608 \zref@addprop\ZREF@mainlist{default}
609 \zref@addprop\ZREF@mainlist{page}

```

#### Mark successful loading

```

610 \let\ZREF@baseok\@empty
611 </base>

```

## 6.3 Module user

```

612 <*user>
613 \NeedsTeXFormat{LaTeX2e}
614 \ProvidesPackage{zref-user}%
615 [2010/04/17 v2.12 Module user for zref (H0)]%
616 \RequirePackage{zref-base}[2010/04/17]
617 \@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.

```

618 \newcommand*\zlabel{%
619   \ifx\label\ltx@gobble
620     \expandafter\ltx@gobble
621   \else
622     \expandafter\zref@wrapper@babel\expandafter\zref@label
623   \fi
624 }%

```

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

```

625 \newcommand*\zref}[2][default]{%
626   \zref@propexists{#1}{%
627     \zref@wrapper@babel\ZREF@zref{#2}{#1}%
628   }%
629 }%
630 \def\ZREF@zref#1{%
631   \zref@refused{#1}%
632   \zref@extract{#1}%
633 }%

```

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

```

634 \newcommand*\zpageref{%
635   \zref[page]%
636 }%

```

**\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.

```

637 \newcommand*\zrefused{\zref@refused}%
638 </user>

```

## 6.4 Module abspage

```

639 <*abspage>
640 \NeedsTeXFormat{LaTeX2e}
641 \ProvidesPackage{zref-abspage}%
642 [2010/04/17 v2.12 Module abspage for zref (H0)]%
643 \RequirePackage{zref-base}[2010/04/17]
644 \@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`.

```

645 \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.

```

646 \begingroup

```

```

647 \let\@addtoreset\ltx@gobbletwo
648 \newcounter{abspage}%
649 \endgroup
650 \setcounter{abspage}{0}%
651 \AtBeginShipout{%
652   \stepcounter{abspage}%
653 }%
654 \zref@newprop*{abspage}[0]{\the\c@abspage}%
655 \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.

```

656 \abspage

```

## 6.5 Module counter

```

657 <*counter>
658 \NeedsTeXFormat{LaTeX2e}
659 \ProvidesPackage{zref-counter}%
660 [2010/04/17 v2.12 Module counter for zref (HO)]%
661 \RequirePackage{zref-base}[2010/04/17]
662 \@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.

```

663 \zref@newprop{counter}{}
664 \zref@addprop\ZREF@mainlist{counter}

```

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

```

665 \AtBeginDocument{%
666   \ZREF@patch{refstepcounter}{%
667     \def\refstepcounter#1{%
668       \zref@setcurrent{counter}{#1}%
669       \ZREF@org@refstepcounter{#1}%
670     }%
671   }%
672 }
673 </counter>

```

## 6.6 Module lastpage

```

674 <*lastpage>
675 \NeedsTeXFormat{LaTeX2e}
676 \ProvidesPackage{zref-lastpage}%
677 [2010/04/17 v2.12 Module lastpage for zref (HO)]%
678 \RequirePackage{zref-base}[2010/04/17]
679 \RequirePackage{zref-abspage}[2010/04/17]
680 \RequirePackage{atveryend}[2009/12/07]
681 \@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.

```

682 \zref@newlist{LastPage}
683 \AfterLastShipout{%
684   \if@filesw
685     \begingroup
686       \advance\c@page\m@ne
687       \toks@\expandafter\expandafter\expandafter{%
688         \expandafter\Z@L@main
689         \Z@L@LastPage

```

```

690     }%
691     \expandafter\zref@wrapper@immediate\expandafter{%
692         \expandafter\ZREF@label\expandafter{\the\toks@}\LastPage}%
693     }%
694     \endgroup
695     \fi
696 }

\zref@iflastpage

697 \def\zref@iflastpage#1{%
698     \ifnum\zref@extractdefault{#1}{abspage}{-1}=%
699         \zref@extractdefault{LastPage}{abspage}{-2} %
700         \expandafter\ltx@firstoftwo
701     \else
702         \expandafter\ltx@secondoftwo
703     \fi
704 }

\ziflastpage

705 \newcommand*{\ziflastpage}{%
706     \zref@wrapper@babel\ZREF@iflastpage
707 }

ZREF@iflastpage

708 \def\ZREF@iflastpage#1{%
709     \zref@refused{LastPage}%
710     \zref@refused{#1}%
711     \zref@iflastpage{#1}%
712 }

713 </lastpage>

```

## 6.7 Module thepage

```

714 <*thepage>
715 \NeedsTeXFormat{LaTeX2e}
716 \ProvidesPackage{zref-thepage}%
717 [2010/04/17 v2.12 Module thepage for zref (HO)]%
718 \RequirePackage{zref-base}[2010/04/17]
719 \@ifundefined{ZREF@baseok}{\endinput}{}

720 \RequirePackage{atbegshi}
721 \RequirePackage{zref-abspage}[2010/04/17]

722 \zref@newlist{thepage}
723 \zref@addprop{thepage}{page}
724 \AtBeginShipout{%
725     \zref@wrapper@immediate{%
726         \zref@labelbylist{thepage\the\value{abspage}}{thepage}%
727     }%
728 }

\zref@thepage@name

729 \ltx@ifundefined{numexpr}{%
730     \def\zref@thepage@name#1{thepage\number#1}%
731 }{%
732     \def\zref@thepage@name#1{thepage\the\numexpr#1}%
733 }

\zref@thepage

734 \def\zref@thepage#1{%
735     \zref@extract{\zref@thepage@name{#1}}{page}%
736 }%

```



\zref@thepage@refused

```
737 \def\zref@thepage@refused#1{%
738   \zref@refused{\zref@thepage@name{#1}}%
739 }%
```

\zthepage

```
740 \newcommand*{\zthepage}[1]{%
741   \zref@thepage@refused{#1}%
742   \zref@thepage{#1}%
743 }
```

```
744 \</thepage>
```

## 6.8 Module nextpage

```
745 \<*nextpage>
746 \NeedsTeXFormat{LaTeX2e}
747 \ProvidesPackage{zref-nextpage}%
748   [2010/04/17 v2.12 Module nextpage for zref (HO)]%
749 \RequirePackage{zref-base}[2010/04/17]
750 \@ifundefined{ZREF@baseok}{\endinput}{}

751 \RequirePackage{zref-abstract}[2010/04/17]
752 \RequirePackage{zref-thepage}[2010/04/17]
753 \RequirePackage{zref-lastpage}[2010/04/17]
754 \RequirePackage{uniquecounter}[2009/12/18]

755 \UniqueCounterNew{znnextpage}
756
757 \newcommand*{\znnextpagesetup}{%
758   \afterassignment\ZREF@np@setup@i
759   \def\ZREF@np@call@unknown##1%
760 }
761 \def\ZREF@np@setup@i{%
762   \afterassignment\ZREF@np@setup@ii
763   \def\ZREF@np@call@nonext##1%
764 }
765 \def\ZREF@np@setup@ii{%
766   \def\ZREF@np@call@next##1%
767 }
768 \def\ZREF@np@call@unknown#1{#1}
769 \def\ZREF@np@call@nonext#1{#1}
770 \def\ZREF@np@call@next#1{#1}
771 \newcommand*{\znnextpage}{%
772   \UniqueCounterCall{znnextpage}{\ZREF@nextpage}%
773 }
774 \newcommand*{\znonextpagename}{}
775 \newcommand*{\zunknownnextpagename}{\Z@D@page}
776 \def\ZREF@nextpage#1{%
777   \begingroup
778     \def\ZREF@refname@this{zref@np#1}%
779     \zref@labelbyprops\ZREF@refname@this{abstract}%
780     \chardef\ZREF@call=0 % unknown
781     \ZREF@ifrefundefined\ZREF@refname@this{%
782       {%
783         \edef\ZREF@pagenum@this{%
784           \zref@extractdefault\ZREF@refname@this{abstract}{0}%
785         }%
786         \edef\ZREF@refname@next{%
787           \zref@thepage@name{%
788             \the\numexpr\ZREF@pagenum@this+1%
789           }%
790         }%
```

```

791 \ifnum\ZREF@pagenum@this>0 %
792 \ZREF@ifrefundefined{LastPage}{%
793 \zref@ifrefundefined\ZREF@refname@next{%
794 }{%
795 \chardef\ZREF@call=2 % next page
796 }%
797 }{%
798 \edef\ZREF@pagenum@last{%
799 \zref@extractdefault{LastPage}{abspage}{0}%
800 }%
801 \ifnum\ZREF@pagenum@this<\ZREF@pagenum@last\ltx@space
802 \ZREF@ifrefundefined\ZREF@refname@next{%
803 }{%
804 \chardef\ZREF@call=2 % next page
805 }%
806 \else
807 \ifnum\ZREF@pagenum@this=\ZREF@pagenum@this\ltx@space
808 \chardef\ZREF@call=1 % no next page
809 \fi
810 \fi
811 }%
812 \fi
813 }%
814 \edef\x{%
815 \endgroup
816 \ifcase\ZREF@call
817 \noexpand\ZREF@np@call@unknown{%
818 \noexpand\zunknownnextpagename
819 }%
820 \or
821 \noexpand\ZREF@np@call@nonext{%
822 \noexpand\znonextpagename
823 }%
824 \else
825 \noexpand\ZREF@np@call@next{%
826 \noexpand\zref@extract{\ZREF@refname@next}{page}%
827 }%
828 \fi
829 }%
830 \x
831 }
832 </nextpage>

```

## 6.9 Module totpages

```

833 <*totpages>
834 \NeedsTeXFormat{LaTeX2e}
835 \ProvidesPackage{zref-totpages}%
836 [2010/04/17 v2.12 Module totpages for zref (H0)]%
837 \RequirePackage{zref-base}[2010/04/17]
838 \@ifundefined{ZREF@baseok}{\endinput}{}

```

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

```

839 \RequirePackage{zref-abspage}[2010/04/17]
840 \RequirePackage{zref-lastpage}[2010/04/17]

```

**\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.

```

841 \newcommand*{\ztotpages}{%
842 \zref@extractdefault{LastPage}{abspage}{0}%
843 }

```

Also we mark the reference `LastPage` as used:

```

844 \AtBeginDocument{%

```

```

845 \zref@refused{LastPage}%
846 }
847 </totpages>

```

## 6.10 Module marks

```

848 <*marks>
849 \NeedsTeXFormat{LaTeX2e}
850 \ProvidesPackage{zref-marks}%
851 [2010/04/17 v2.12 Module marks for zref (H0)]%
852 \RequirePackage{zref-base}[2010/04/17]
853 \@ifundefined{ZREF@baseok}{\endinput}{}

854 \RequirePackage{kvsetkeys}[2009/07/30]
855 \newcommand*{\zref@marks@register}[3][{}]{%
856   \edef\ZREF@TempName{#1}%
857   \edef\ZREF@TempNum{\ZREF@number#2}%
858   \ifnum\ZREF@TempNum<\ltx@zero %
859     \PackageError\ZREF@name{%
860       \string\zref@marks@register\ltx@space is called with invalid%
861       \MessageBreak
862       marks register number (\ZREF@TempNum)%
863     }{%
864       Use '0' or the command, defined by \string\newmarks.\MessageBreak
865       \@ehc
866     }%
867   \else
868     \ifx\ZREF@TempName\ltx@empty
869       \edef\ZREF@TempName{mark\romannumeral\ZREF@TempNum}%
870     \else
871       \edef\ZREF@TempName{marks\ZREF@TempName}%
872     \fi
873     \ZREF@MARKS@DefineProp{top}%
874     \ZREF@MARKS@DefineProp{first}%
875     \ZREF@MARKS@DefineProp{bot}%
876     \kv@parse{#3}{%
877       \ifx\kv@value\relax
878         \def\kv@value{top,first,bot}%
879       \fi
880       \edef\ZREF@temp{\expandafter\@car\kv@key X\@nil}%
881       \ifx\ZREF@temp\ZREF@STAR
882         \edef\kv@key{\expandafter\@cdr\kv@key\@nil}%
883       \zref@newlist\kv@key
884     \fi
885     \expandafter\comma@parse\expandafter{\kv@value}{%
886       \ifcase0\ifx\comma@entry\ZREF@NAME@top 1\else
887         \ifx\comma@entry\ZREF@NAME@first 1\else
888         \ifx\comma@entry\ZREF@NAME@bot 1\fi\fi\fi\ltx@space
889       \PackageWarning\ZREF@name{%
890         Use 'top', 'first' or 'bot' for the list values%
891         \MessageBreak
892         in the third argument of \string\zref@marks@register.%
893         \MessageBreak
894         Ignoring unkown value '\comma@entry'%
895       }%
896     \else
897       \zref@addprop{\kv@key}{\comma@entry\ZREF@TempName}%
898     \fi
899     \ltx@gobble
900   }%
901   \ltx@gobbletwo
902 }%
903 \fi

```

```

904 }
905 \def\ZREF@STAR{*}
906 \def\ZREF@NAME@top{top}
907 \def\ZREF@NAME@first{first}
908 \def\ZREF@NAME@bot{bot}
909 \def\ZREF@MARKS@DefineProp#1{%
910   \zref@ifpropundefined{#1\ZREF@TempName}{%
911     \ifnum\ZREF@TempNum=\lt x@zero
912       \begingroup
913         \edef\x{\endgroup
914           \noexpand\zref@newprop*{#1\ZREF@TempName}[]{%
915             \expandafter\noexpand\csname#1mark\endcsname
916           }%
917         }%
918       \x
919     \else
920       \begingroup
921       \edef\x{\endgroup
922         \noexpand\zref@newprop*{#1\ZREF@TempName}[]{%
923           \expandafter\noexpand\csname#1marks\endcsname
924           \ZREF@TempNum
925         }%
926       }%
927       \x
928     \fi
929   }{%
930     \PackageWarning\ZREF@name{%
931       \string\zref@marks@register\lt x@space does not generate the%
932       \MessageBreak
933       new property ‘#1\ZREF@TempName’, because\MessageBreak
934       it is already defined%
935     }%
936   }%
937 }
938 \</marks>

```

## 6.11 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.

```

939 \< *runs>
940 \NeedsTeXFormat{LaTeX2e}
941 \ProvidesPackage{zref-runs}%
942 [2010/04/17 v2.12 Module runs for zref (H0)]%

```

\zruns

```

943 \providecommand*\zruns{0}%
944 \AtBeginDocument{%
945   \edef\zruns{\number\numexpr\zruns+1}%
946   \begingroup
947     \def\on@line{}%
948     \PackageInfo{zref-runs}{LaTeX runs: \zruns}%
949     \if@filesw
950       \immediate\write\@mainaux{%
951         \string\gdef\string\zruns{\zruns}%
952       }%
953     \fi
954   \endgroup
955 }
956 \</runs>

```

## 6.12 Module `perpage`

```

957 <*perpage>
958 \NeedsTeXFormat{LaTeX2e}
959 \ProvidesPackage{zref-perpage}%
960 [2010/04/17 v2.12 Module perpage for zref (HO)]%
961 \RequirePackage{zref-base}[2010/04/17]
962 \@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.

```

963 \RequirePackage{zref-abspage}[2010/04/17]

```

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

```

964 \zref@newprop*{pagevalue}[0]{\number\c@page}
965 \zref@newlist{perpage}
966 \zref@addprop{perpage}{abspage}
967 \zref@addprop{perpage}{page}
968 \zref@addprop{perpage}{pagevalue}

```

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

```

969 \newcounter{zpage}

```

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

```

970 \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`.

```

971 \let\ZREF@org@stepcounter\stepcounter
972 \def\stepcounter#1{%
973   \ifcsname @stepcounterhook@#1\endcsname
974     \csname @stepcounterhook@#1\endcsname
975   \fi
976   \ZREF@org@stepcounter{#1}%
977 }

```

**\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.

```

978 \newcommand*{\zmakeperpage}{%
979   \@ifnextchar[\ZREF@makeperpage@opt{\ZREF@@makeperpage[\z@]}%
980 }

```

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

```

981 \def\ZREF@makeperpage@opt[#1]{%
982   \begingroup
983     \edef\x{\endgroup
984       \noexpand\ZREF@@makeperpage[\number\numexpr#1-1\relax]%
985     }%
986   \x
987 }

988 \def\ZREF@@makeperpage[#1]#2{%
989   \@ifundefined{@stepcounterhook@#2}{%
990     \expandafter\gdef\csname @stepcounterhook@#2\endcsname{%
991     }%
992     \expandafter\gdef\csname ZREF@perpage@#2\endcsname{%
993       \ZREF@@perpage@step{#2}{#1}%
994     }%

```

```

995 \expandafter\g@addto@macro\csname @stepcounterhook@#2\endcsname{%
996   \ifcsname ZREF@perpage@#2\endcsname
997     \csname ZREF@perpage@#2\endcsname
998   \fi
999 }%
1000 }

```

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

```

1001 \def\ZREF@perpage@step#1#2{%
First the reference is generated.
1002 \global\advance\c@zref@unique\@ne
1003 \begingroup
1004 \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.

```

1005 \zref@ifrefundefined\thezref@unique{%
1006   \global\c@zpage=\c@page
1007   \global\let\thezpage\thepage
1008   \expandafter\xdef\csname ZREF@abspage@#1\endcsname{\number\c@abspage}%
1009 }{%

```

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

```

1010 \global\c@zpage=\zref@extract\thezref@unique{pagevalue}\relax
1011 \xdef\thezpage{\noexpand\zref@extract{\thezref@unique}{page}}%
1012 \expandafter\xdef\csname ZREF@abspage@#1\endcsname{%
1013   \zref@extractdefault\thezref@unique{abspage}{\number\c@abspage}%
1014 }%
1015 }%

```

Page changes are detected by a changed absolute page number.

```

1016 \expandafter\ifx\csname ZREF@abspage@#1\expandafter\endcsname
1017   \csname ZREF@currentabspage@#1\endcsname
1018 \else
1019   \global\csname c@#1\endcsname=#2\relax
1020   \global\expandafter\let
1021     \csname ZREF@currentabspage@#1\expandafter\endcsname
1022     \csname ZREF@abspage@#1\endcsname
1023 \fi
1024 \endgroup
1025 }

```

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

```

1026 \newcommand*{\zunmakeperpage}[1]{%
1027   \global\expandafter\let\csname ZREF@perpage@#1\endcsname\@undefined
1028 }
1029 </perpage>

```

## 6.13 Module `titleref`

```

1030 <*titleref>
1031 \NeedsTeXFormat{LaTeX2e}
1032 \ProvidesPackage{zref-titleref}%
1033 [2010/04/17 v2.12 Module titleref for zref (HO)]%
1034 \RequirePackage{zref-base}[2010/04/17]
1035 \@ifundefined{ZREF@baseok}{\endinput}{}
1036 \RequirePackage{getttitlestring}[2009/12/08]

```

### 6.13.1 Implementation

1037 \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`.

1038 \let\zref@titleref@current\@empty

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

1039 \zref@newprop{title}{\zref@titleref@current}%

1040 \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.

`\ifzref@titleref@expand` The switch `\ifzref@titleref@expand` distinguishes between the both methods. Package `nameref`'s behaviour is achieved by setting the switch to false, otherwise `titleref`'s expansion is used. Default is false.

1041 \newif\ifzref@titleref@expand

`\ZREF@titleref@hook` The hook `\ZREF@titleref@hook` 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 `\zref@titleref@current`.

1042 \let\ZREF@titleref@hook\@empty

`\zref@titleref@cleanup` The hook should not be used directly, instead we provide the macro `\zref@titleref@cleanup` to add stuff to the hook and prevents that a previous non-empty content is not discarded accidentally.

1043 \def\zref@titleref@cleanup#1{%

1044   \begingroup

1045   \toks@ \expandafter{%

1046     \ZREF@titleref@hook

1047     #1%

1048   }%

1049   \expandafter\endgroup

1050   \expandafter\def\expandafter\ZREF@titleref@hook\expandafter{%

1051     \the\toks@

1052   }%

1053 }%

`\ifzref@titleref@stripperperiod` Sometimes a title contains a period at the end. Package `nameref` removes this. This behaviour is controlled by the switch `\ifzref@titleref@stripperperiod` and works regardless of the setting of option `expand`. Period stripping is the default.

1054 \newif\ifzref@titleref@stripperperiod

1055 \zref@titleref@stripperperiodtrue

`\zref@titleref@setcurrent` Macro `\zref@titleref@setcurrent` sets a new current title stored in `\zref@titleref@current`. Some cleanup and expansion is performed that can be controlled by the previous switches.

1056 \def\zref@titleref@setcurrent#1{%

1057   \ifzref@titleref@expand

1058     \GetTitleStringExpand{#1}%

1059   \else

1060     \GetTitleStringNonExpand{#1}%

1061   \fi

1062   \edef\zref@titleref@current{%

```

1063 \detokenize\expandafter{\GetTitleStringResult}%
1064 }%
1065 \ifzref@titleref@stripperperiod
1066 \edef\zref@titleref@current{%
1067 \expandafter\ZREF@stripperperiod\zref@titleref@current
1068 \@empty.\@empty\@nil
1069 }%
1070 \fi
1071 }%
1072 \GetTitleStringDisableCommands{%
1073 \ZREF@titleref@hook
1074 }

```

`\ZREF@stripperperiod` If `\ZREF@stripperperiod` is called, the argument consists of space tokens and tokens with catcode 12 (other), because of  $\varepsilon$ -TEX's `\detokenize`.

```

1075 \def\ZREF@stripperperiod#1.\@empty#2\@nil{#1}%

```

### 6.13.2 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`.

```

1076 \define@key{ZREF@TR}{expand}[true]{%
1077 \csname zref@titleref@expand#1\endcsname
1078 }%
1079 \define@key{ZREF@TR}{stripperperiod}[true]{%
1080 \csname zref@titleref@stripperperiod#1\endcsname
1081 }%
1082 \define@key{ZREF@TR}{cleanup}{%
1083 \zref@titleref@cleanup{#1}%
1084 }%
1085 \define@key{ZREF@TR}{title}{%
1086 \def\zref@titleref@current{#1}%
1087 }%
1088 \newcommand*{\ztitlerefsetup}{%
1089 \setkeys{ZREF@TR}%
1090 }%

```

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

```

1091 \newcommand*{\ztitleref}{%
1092 \zref@wrapper@babel\ZREF@titleref
1093 }%
1094 \def\ZREF@titleref#1{%
1095 \begingroup
1096 \zref@refused{#1}%
1097 \let\label\ltx@gobble
1098 \zref@extract{#1}{title}%
1099 \endgroup
1100 }%

```

### 6.13.3 Patches for section and caption commands

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

Captions of figures and tables.

```

1101 \AtBeginDocument{%
1102 \ZREF@patch{@caption}{%
1103 \long\def\@caption#1[#2]{%
1104 \zref@titleref@setcurrent{#2}%
1105 \ZREF@org@@caption{#1}[{#2}]%
1106 }%
1107 }%

```



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

```

1108 \ZREF@patch{@part}{%
1109   \def\@part[#1]{%
1110     \zref@titleref@setcurrent{#1}%
1111     \ZREF@org@@part[{#1}]%
1112   }%
1113 }%
1114 \ZREF@patch{@chapter}{%
1115   \def\@chapter[#1]{%
1116     \zref@titleref@setcurrent{#1}%
1117     \ZREF@org@@chapter[{#1}]%
1118   }%
1119 }%
1120 \ZREF@patch{@sect}{%
1121   \def\@sect#1#2#3#4#5#6[#7]{%
1122     \zref@titleref@setcurrent{#7}%
1123     \ZREF@org@@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}]%
1124   }%
1125 }%

```

The star versions of the section commands.

```

1126 \ZREF@patch{@spart}{%
1127   \def\@spart#1{%
1128     \zref@titleref@setcurrent{#1}%
1129     \ZREF@org@@spart{#1}%
1130   }%
1131 }%
1132 \ZREF@patch{@schapter}{%
1133   \def\@schapter#1{%
1134     \zref@titleref@setcurrent{#1}%
1135     \ZREF@org@@schapter{#1}%
1136   }%
1137 }%
1138 \ZREF@patch{@ssect}{%
1139   \def\@ssect#1#2#3#4#5{%
1140     \zref@titleref@setcurrent{#5}%
1141     \ZREF@org@@ssect{#1}{#2}{#3}{#4}{#5}%
1142   }%
1143 }%

```

#### 6.13.4 Class memoir

```

1144 \@ifclassloaded{memoir}{%
1145   \ltx@ifundefined{ifheadnameref}{}{%
1146     \def\@chapter[#1]#2{%
1147       \ltx@ifundefined{ch@pt@c}{%
1148         \zref@titleref@setcurrent{#1}%
1149       }{%
1150         \ifx\ch@pt@c\ltx@empty
1151           \zref@titleref@setcurrent{#2}%
1152         \else
1153           \def\NR@temp{#1}%
1154           \ifx\NR@temp\ltx@empty
1155             \expandafter\zref@titleref@setcurrent
1156             \expandafter{\ch@pt@c}%
1157           \else
1158             \ifheadnameref
1159               \zref@titleref@setcurrent{#1}%
1160             \else
1161               \expandafter\zref@titleref@setcurrent
1162               \expandafter{\ch@pt@c}%
1163             \fi

```

```

1164         \fi
1165     \fi
1166 }%
1167 \ZREF@org@@chapter[{#1}]{#2}%
1168 }%
1169 \ZREF@patch{M@sect}{%
1170     \def\M@sect#1#2#3#4#5#6[#7][#8]{%
1171         \ifheadnameref
1172             \zref@titleref@setcurrent{#8}%
1173         \else
1174             \zref@titleref@setcurrent{#7}%
1175         \fi
1176         \ZREF@org@M@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}][{#8}]%
1177     }%
1178 }%
1179 }%
1180 }{}%

```

### 6.13.5 Class beamer

```

1181 \@ifclassloaded{beamer}{%
1182     \ZREF@patch{beamer@section}{%
1183         \long\def\beamer@section[#1]{%
1184             \zref@titleref@setcurrent{#1}%
1185             \ZREF@org@beamer@section[{#1}]%
1186         }%
1187     }%
1188     \ZREF@patch{beamer@subsection}{%
1189         \long\def\beamer@subsection[#1]{%
1190             \zref@titleref@setcurrent{#1}%
1191             \ZREF@org@beamer@subsection[{#1}]%
1192         }%
1193     }%
1194     \ZREF@patch{beamer@subsubsection}{%
1195         \long\def\beamer@subsubsection[#1]{%
1196             \zref@titleref@setcurrent{#1}%
1197             \ZREF@org@beamer@subsubsection[{#1}]%
1198         }%
1199     }%
1200 }{}%

```

### 6.13.6 Package titlesec

```

1201 \@ifpackageloaded{titlesec}{%
1202     \ZREF@patch{ttl@sect@i}{%
1203         \def\ttl@sect@i#1#2[#3]#4{%
1204             \zref@titleref@setcurrent{#4}%
1205             \ZREF@org@ttl@sect@i{#1}{#2}[{#3}]{#4}%
1206         }%
1207     }%
1208 }{}%

```

### 6.13.7 Package longtable

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

```

1209 \@ifpackageloaded{longtable}{%
1210     \ZREF@patch{LT@c@ption}{%
1211         \def\LT@c@ption#1[#2]#3{%
1212             \ZREF@org@LT@c@ption{#1}[{#2}]{#3}%
1213             \zref@titleref@setcurrent{#2}%
1214         }%
1215     }%
1216 }{}%

```

### 6.13.8 Package listings

Package listings: support for its caption.

```

1217 \@ifpackageloaded{listings}{%
1218   \ZREF@patch{lst@MakeCaption}{%
1219     \def\lst@MakeCaption{%
1220       \ifx\lst@label\@empty
1221       \else
1222         \expandafter\zref@titleref@setcurrent\expandafter{%
1223           \lst@caption
1224         }%
1225       \fi
1226       \ZREF@org@lst@MakeCaption
1227     }%
1228   }%
1229 }{}%
```

### 6.13.9 Theorems

```

1230 \ZREF@patch{@opargbegintheorem}{%
1231   \def\@opargbegintheorem#1#2#3{%
1232     \zref@titleref@setcurrent{#3}%
1233     \ZREF@org@@opargbegintheorem{#1}{#2}{#3}%
1234   }%
1235 }%
1236 \@ifpackageloaded{amsthm}{%
1237   \begingroup
1238     \edef\x{macro:\string#1\string#2[\string#3]}%
1239     \@onelevel@sanitize\x
1240     \def\y#1->#2\@nil{#1}%
1241     \edef\z{\expandafter\y\meaning\@begintheorem->\@nil}%
1242     \@onelevel@sanitize\z
1243   \expandafter\endgroup
1244   \ifx\x\z
1245     \ZREF@patch{@begintheorem}{%
1246       \def\@begintheorem#1#2[#3]{%
1247         \zref@titleref@setcurrent{#3}%
1248         \ZREF@org@@begintheorem{#1}{#2}[%3]}%
1249     }%
1250   }%
1251   \fi
1252 }{}%
1253 }
1254 </titleref>
```

## 6.14 Module xr

```

1255 <*xr>
1256 \NeedsTeXFormat{LaTeX2e}
1257 \ProvidesPackage{zref-xr}%
1258   [2010/04/17 v2.12 Module xr for zref (HO)]%
1259 \RequirePackage{zref-base}[2010/04/17]
1260 \@ifundefined{ZREF@baseok}{\endinput}{}

1261 \RequirePackage{keyval}
1262 \RequirePackage{kvoptions}[2010/02/22]
```

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.

```

1263 \zref@newprop{url}{}%
1264 \zref@newprop{externaldocument}{}%
```

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.

```
1265 \providecommand*\zref@xr@ext{%
1266   \ltx@ifundefined{XR@ext}{pdf}{\XR@ext}%
1267 }%
```

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

```
1268 \newif\ifZREF@xr@zreflabel

1269 \SetupKeyvalOptions{%
1270   family=ZREF@XR,%
1271   prefix=ZREF@xr@%
1272 }
1273 \DeclareBoolOption[true]{tozreflabel}
1274 \DeclareBoolOption[false]{toltxlabel}
1275 \DeclareBoolOption{verbose}
1276 \define@key{ZREF@XR}{ext}{%
1277   \def\zref@xr@{#1}%
1278 }
```

`\zxrsetup`

```
1279 \newcommand*\zxrsetup{%
1280   \setkeys{ZREF@XR}%
1281 }%
```

`\xexternaldocument` 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).

```
1282 \newcommand*\xexternaldocument{%
1283   \zref@ifpropundefined{title}{%
1284     \zref@newprop{title}{}%
1285   }{}%
1286   \zref@ifpropundefined{anchor}{%
1287     \zref@newprop{anchor}{%
1288       \ltx@ifundefined{@currentHref}{-}{\@currentHref}%
1289     }%
1290   }{}%
1291   \zref@ifpropundefined{url}{%
1292     \zref@newprop{url}{}%
1293   }{}%
1294   \begingroup
1295     \csname @safe@actives@true\endcsname
1296     \makeatletter
1297     \@ifstar{%
1298       \ZREF@xr@zreflabelfalse
1299       \@testopt\ZREF@xr@externaldocument{}%
1300     }{%
1301       \ZREF@xr@zreflabeltrue
1302       \@testopt\ZREF@xr@externaldocument{}%
1303     }%
1304   }%
```

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.

```

1305 \def\ZREF@xr@externaldocument[#1]#2{%
1306     \def\ZREF@xr@prefix{#1}%
1307     \let\ZREF@xr@filelist\@empty
1308     \edef\ZREF@xr@externalfile{#2}%
1309     \edef\ZREF@xr@file{\ZREF@xr@externalfile.aux}%
1310     \filename@parse{#2}%
1311     \@testopt\ZREF@xr@graburl{#2.\zref@xr@ext}%
1312 }%
1313 \def\ZREF@xr@graburl[#1]{%
1314     \edef\ZREF@xr@url{#1}%
1315     \ZREF@xr@checkfile
1316 \endgroup
1317 }%

```

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

```

1318 \def\ZREF@xr@checkfile{%
1319     \openin\@inputcheck\ZREF@xr@file\relax
1320     \ifeof\@inputcheck
1321         \PackageWarning{zref-xr}{%
1322             File '\ZREF@xr@file' not found or empty,\MessageBreak
1323             labels not imported%
1324         }%
1325     \else
1326         \PackageInfo{zref-xr}{%
1327             Label \ifZREF@xr@zreflabel (zref) \fi import from '\ZREF@xr@file'%
1328         }%
1329         \def\ZREF@xr@found{0}%
1330         \def\ZREF@xr@ignored@empty{0}%
1331         \def\ZREF@xr@ignored@zref{0}%
1332         \def\ZREF@xr@ignored@ltx{0}%
1333         \ZREF@xr@processfile
1334         \closein\@inputcheck
1335         \begingroup
1336         \let\on@line\@empty
1337         \PackageInfo{zref-xr}{%
1338             Statistics for '\ZREF@xr@file':\MessageBreak
1339             \ZREF@xr@found\space
1340             \ifZREF@xr@zreflabel zref\else LaTeX\fi\space
1341             label(s) found%
1342             \ifnum\ZREF@xr@ignored@empty>0 %
1343                 ,\MessageBreak
1344                 \ZREF@xr@ignored@empty\space empty label(s) ignored%
1345             \fi
1346             \ifnum\ZREF@xr@ignored@zref>0 %
1347                 ,\MessageBreak
1348                 \ZREF@xr@ignored@zref\space duplicated zref label(s) ignored%
1349             \fi
1350             \ifnum\ZREF@xr@ignored@ltx>0 %
1351                 ,\MessageBreak
1352                 \ZREF@xr@ignored@ltx\space duplicated latex label(s) ignored%
1353             \fi
1354         }%
1355     \endgroup
1356 \fi
1357 \ifx\ZREF@xr@filelist\@empty
1358 \else
1359     \edef\ZREF@xr@file{\expandafter\@car\ZREF@xr@filelist\@nil}%
1360     \edef\ZREF@xr@filelist{\expandafter\@cdr\ZREF@xr@filelist\@nil}%
1361     \expandafter\ZREF@xr@checkfile
1362 \fi
1363 }%

```

\ZREF@xr@processfile

```
1364 \def\ZREF@xr@processfile{%
1365   \read\@inputcheck to\ZREF@xr@line
1366   \expandafter\ZREF@xr@processline\ZREF@xr@line..\ZREF@nil
1367   \ifeof\@inputcheck
1368   \else
1369     \expandafter\ZREF@xr@processfile
1370   \fi
1371 }%
```

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

```
1372 \long\def\ZREF@xr@processline#1#2#3\ZREF@nil{%
1373   \def\x{#1}%
1374   \toks@{#2}%
1375   \ifZREF@xr@zreflabel
1376     \ifx\x\ZREF@xr@zref@newlabel
1377       \expandafter\ZREF@xr@process@zreflabel\ZREF@xr@line...\ZREF@nil
1378     \fi
1379   \else
1380     \ifx\x\ZREF@xr@newlabel
1381       \expandafter\ZREF@xr@process@label\ZREF@xr@line...[]\ZREF@nil
1382     \fi
1383   \fi
1384   \ifx\x\ZREF@xr@@input
1385     \edef\ZREF@xr@filelist{%
1386       \etex@unexpanded\expandafter{\ZREF@xr@filelist}%
1387       {\filename@area\the\toks@}%
1388     }%
1389   \fi
1390 }%
1391 \def\ZREF@xr@process@zreflabel\zref@newlabel#1#2#3\ZREF@nil{%
1392   \edef\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1393   \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1394   \def\x{#2}%
1395   \edef\ZREF@xr@tempname{$temp$}%
1396   \edef\ZREF@xr@temprefname{Z@R@\ZREF@xr@tempname}%
1397   \let\ZREF@xr@list\x
1398   \ifx\ZREF@xr@list\@empty
1399     \PackageWarningNoLine{zref-xr}{%
1400       Label ‘#1’ without properties ignored\MessageBreak
1401       in file ‘\ZREF@xr@file’%
1402     }%
1403     \edef\ZREF@xr@ignored@empty{%
1404       \the\numexpr\ZREF@xr@ignored@empty+1\relax
1405     }%
1406   \else
1407     \expandafter\ZREF@xr@checklist\x\ZREF@nil
1408     \expandafter\let\csname\ZREF@xr@temprefname\endcsname\x
1409     \expandafter\ltx@LocalAppendToMacro
1410     \csname\ZREF@xr@temprefname\endcsname
1411     \expandafter{%
1412       \expandafter\externaldocument\expandafter{%
1413         \ZREF@xr@externalfile
1414       }%
1415     }%
1416     \ZREF@xr@urlcheck\ZREF@xr@tempname
1417     \ifZREF@xr@tozreflabel
1418       \ifundefined{\ZREF@xr@refname}{%
1419         \ifZREF@xr@verbose
1420           \PackageInfo{zref-xr}{%
1421             Import to zref label ‘\ZREF@xr@tempname#1’%
1422           }%

```

```

1423     \fi
1424     \global\expandafter
1425     \let\csname\ZREF@xr@refname\expandafter\endcsname
1426     \csname\ZREF@xr@temprefname\endcsname
1427   }{%
1428     \ZREF@xr@zref@ignorewarning{\ZREF@xr@prefix#1}%
1429   }%
1430   \fi
1431   \ifZREF@xr@toltxlabel
1432     \ZREF@xr@tolabel{\ZREF@xr@tempname}{\ZREF@xr@prefix#1}%
1433   \fi
1434   \fi
1435 }%
1436 \def\ZREF@xr@process@label\newlabel#1#2#3[#4]#5\ZREF@nil{%
1437   \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1438   \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1439   \def\x{#2}%
1440   \edef\ZREF@xr@tempname{$temp$}%
1441   \edef\ZREF@xr@temprefname{Z@R@\ZREF@xr@tempname}%
1442   \expandafter\ZREF@xr@scanparams
1443     \csname\ZREF@xr@temprefname\expandafter\endcsname
1444     \x{}{}{}{}{}\ZREF@nil
1445   \ifx\\#4\\%
1446   \else
1447     % ntheorem knows an optional argument at the end of \newlabel
1448     \zref@ifpropundefined{theotype}{%
1449       \zref@newprop{theotype}{}%
1450     }{}%
1451     \expandafter\ltx@LocalAppendToMacro
1452       \csname\ZREF@xr@temprefname\endcsname{\theotype{#4}}%
1453   \fi
1454   \expandafter\ltx@LocalAppendToMacro
1455     \csname\ZREF@xr@temprefname\expandafter\endcsname\expandafter{%
1456     \expandafter\externaldocument\expandafter{%
1457       \ZREF@xr@externalfile
1458     }%
1459   }%
1460   \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1461   \ifZREF@xr@tozreflabel
1462     \@ifundefined{\ZREF@xr@refname}{%
1463       \ifZREF@xr@verbose
1464         \PackageInfo{zref-xr}{%
1465           Import to zref label ‘\ZREF@xr@prefix#1’%
1466         }%
1467       \fi
1468       \global\expandafter
1469       \let\csname\ZREF@xr@refname\expandafter\endcsname
1470       \csname\ZREF@xr@temprefname\endcsname
1471     }{%
1472       \ZREF@xr@zref@ignorewarning{\ZREF@xr@prefix#1}%
1473     }%
1474   \fi
1475   \ifZREF@xr@toltxlabel
1476     \ZREF@xr@tolabel{\ZREF@xr@tempname}{\ZREF@xr@prefix#1}%
1477   \fi
1478 }
1479 \def\ZREF@xr@zref@newlabel{\zref@newlabel}%
1480 \def\ZREF@xr@newlabel{\newlabel}%
1481 \def\ZREF@xr@@input{\@input}%
1482 \def\ZREF@xr@relax{\relax}%

```

\ZREF@xr@tolabel

```

1483 \def\ZREF@xr@tolabel#1#2{%
1484   \ifZREF@xr@verbose
1485     \PackageInfo{zref-xr}{%
1486       Import to LaTeX label ‘#2’%
1487     }%
1488   \fi
1489   \expandafter\xdef\csname r@#2\endcsname{%
1490     {%
1491       \ltx@ifundefined{M@TitleReference}{%
1492         \ltx@ifundefined{TR@TitleReference}{%
1493           \zref@extractdefault{#1}{default}{}%
1494         }{%
1495           \noexpand\TR@TitleReference
1496             {\zref@extractdefault{#1}{default}{}}%
1497             {\zref@extractdefault{#1}{title}{}}%
1498         }%
1499       }{%
1500         \noexpand\M@TitleReference
1501             {\zref@extractdefault{#1}{default}{}}%
1502             {\zref@extractdefault{#1}{title}{}}%
1503       }%
1504     }%
1505     {\zref@extractdefault{#1}{page}{}}%
1506     \ltx@ifpackageloaded{nameref}{%
1507       {\zref@extractdefault{#1}{title}{}}%
1508       {\zref@extractdefault{#1}{anchor}{}}%
1509       {\zref@extractdefault{#1}{url}{}}%
1510     }{}%
1511   }%
1512 }

```

\ZREF@xr@zref@ignorewarning

```

1513 \def\ZREF@xr@zref@ignorewarning#1{%
1514   \PackageWarningNoLine{zref-xr}{%
1515     Zref label ‘#1’ is already in use\MessageBreak
1516     in file ‘\ZREF@xr@file’%
1517   }%
1518   \edef\ZREF@xr@ignored@empty{%
1519     \the\numexpr\ZREF@xr@ignored+1%
1520   }%
1521 }%

```

\ZREF@xr@ltx@ignorewarning

```

1522 \def\ZREF@xr@ltx@ignorewarning#1{%
1523   \PackageWarningNoLine{zref-xr}{%
1524     LaTeX label ‘#1’ is already in use\MessageBreak
1525     in file ‘\ZREF@xr@file’%
1526   }%
1527   \edef\ZREF@xr@ignored@ltx{%
1528     \the\numexpr\ZREF@xr@ignored@ltx+1%
1529   }%
1530 }%

```

\ZREF@xr@checklist

```

1531 \def\ZREF@xr@checklist#1#2#3\ZREF@nil{%
1532   \ifx@undefined#1\relax
1533     \expandafter\ZREF@xr@checkkey\string#1\@nil
1534   \fi
1535   \ifx\#3\%
1536   \else
1537     \ltx@ReturnAfterFi{%
1538       \ZREF@xr@checklist#3\ZREF@nil

```



```

1539 }%
1540 \fi
1541 }%
1542 \def\ZREF@xr@checkkey#1#2\@nil{%
1543 \zref@ifpropundefined{#2}{%
1544 \zref@newprop{#2}{}%
1545 }{}%
1546 }%

```

\ZREF@xr@scanparams

```

1547 \def\ZREF@xr@scanparams#1#2#3#4#5#6#7\ZREF@nil{%
1548 \let#1\@empty
1549 \ZREF@foundfalse
1550 \ZREF@xr@scantitleref#1#2\TR@TitleReference{}{}\ZREF@nil
1551 \ifZREF@found
1552 \else
1553 \ltx@LocalAppendToMacro#1{\default{#2}}%
1554 \fi
1555 % page
1556 \ltx@LocalAppendToMacro#1{\page{#3}}%
1557 % nameref title
1558 \ifZREF@found
1559 \else
1560 \ifx\\#4\\%
1561 \else
1562 \def\ZREF@xr@temp{#4}%
1563 \ifx\ZREF@xr@temp\ZREF@xr@relax
1564 \else
1565 \ltx@LocalAppendToMacro#1{\title{#4}}%
1566 \fi
1567 \fi
1568 \fi
1569 % anchor
1570 \ifx\\#5\\%
1571 \else
1572 \ltx@LocalAppendToMacro#1{\anchor{#5}}%
1573 \fi
1574 \ifx\\#6\\%
1575 \else
1576 \ltx@LocalAppendToMacro#1{\url{#6}}%
1577 \fi
1578 }%

```

\ZREF@xr@scantitleref

```

1579 \def\ZREF@xr@scantitleref#1#2\TR@TitleReference#3#4#5\ZREF@nil{%
1580 \ifx\\#5\\%
1581 \else
1582 \ltx@LocalAppendToMacro#1{%
1583 \default{#3}%
1584 \title{#4}%
1585 }%
1586 \ZREF@foundtrue
1587 \fi
1588 }%

```

\ZREF@xr@urlcheck

```

1589 \def\ZREF@xr@urlcheck#1{%
1590 \zref@ifrefcontainsprop{#1}{anchor}{%
1591 \zref@ifrefcontainsprop{#1}{url}{%
1592 }{}%
1593 \expandafter
1594 \ltx@LocalAppendToMacro\csname Z@R@#1\expandafter\endcsname

```

```

1595     \expandafter{%
1596     \expandafter\url\expandafter{\ZREF@xr@url}}%
1597     }%
1598     }%
1599   }{%
1600   }%
1601 }%

1602 </xr>

```

## 6.15 Module hyperref

UNFINISHED :-(

```

1603 <*hyperref>
1604 \NeedsTeXFormat{LaTeX2e}
1605 \ProvidesPackage{zref-hyperref}%
1606 [2010/04/17 v2.12 Module hyperref for zref (H0)]%
1607 \RequirePackage{zref-base}[2010/04/17]
1608 \@ifundefined{ZREF@baseok}{\endinput}{}

1609 \zref@newprop{anchor}[]{}%
1610 \ltx@ifundefined{@currentHref}{}{\@currentHref}%
1611 }%
1612 \zref@addprop\ZREF@mainlist{anchor}%
1613 </hyperref>

```

## 6.16 Module savepos

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

### 6.16.1 Identification

```

1614 <*savepos>
1615 \NeedsTeXFormat{LaTeX2e}
1616 \ProvidesPackage{zref-savepos}%
1617 [2010/04/17 v2.12 Module savepos for zref (H0)]%
1618 \RequirePackage{zref-base}[2010/04/17]
1619 \@ifundefined{ZREF@baseok}{\endinput}{}

```

### 6.16.2 Availability

First we check, whether the feature is available.

```

1620 \ltx@ifundefined{pdfsavepos}{%
1621   \PackageError\ZREF@name{%
1622     \string\pdfsavepos\space is not supported.\MessageBreak
1623     It is provided by pdfTeX (1.40) or XeTeX%
1624   }\ZREF@UpdatePdfTeX
1625   \endinput
1626 }{}%

```

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.

```

1627 \RequirePackage{ifpdf}
1628 \ifpdf
1629 \else
1630   \begingroup\expandafter\expandafter\expandafter\endgroup
1631   \expandafter\ifx\csname pdftexversion\endcsname\relax
1632   \else
1633     \ifnum\pdftexversion<140 %
1634     \PackageError\ZREF@name{%
1635       \string\pdfsavepos\space is not supported in DVI mode\MessageBreak

```

```

1636         of this pdfTeX version%
1637     }\ZREF@UpdatePdfTeX
1638     \expandafter\expandafter\expandafter\endinput
1639 \fi
1640 \fi
1641 \fi

```

### 6.16.3 Setup

```

1642 \zref@newlist{savepos}
1643 \zref@newprop*{posx}[0]{\the\pdflastxpos}
1644 \zref@newprop*{posy}[0]{\the\pdflastypos}
1645 \zref@addprop{savepos}{posx}
1646 \zref@addprop{savepos}{posy}

```

### 6.16.4 User macros

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

```

1647 \def\zsavepos#1{%
1648   \@bsphack
1649   \if@filesw
1650     \pdfsavepos
1651     \zref@labelbylist{#1}{savepos}%
1652   \fi
1653   \@esphack
1654 }

```

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

```

1655 \newcommand*{\zposx}[1]{%
1656   \zref@extract{#1}{posx}%
1657 }%
1658 \newcommand*{\zposy}[1]{%
1659   \zref@extract{#1}{posy}%
1660 }%

```

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.

```

1661 \</savepos>

```

## 6.17 Module dotfill

```

1662 \<*dotfill>
1663 \NeedsTeXFormat{LaTeX2e}
1664 \ProvidesPackage{zref-dotfill}%
1665   [2010/04/17 v2.12 Module dotfill for zref (HO)]%
1666 \RequirePackage{zref-base}[2010/04/17]
1667 \@ifundefined{ZREF@baseok}{\endinput}{\<}

```

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

```

1668 \RequirePackage{zref-savepos}[2010/04/17]

```

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

```

1669 \zref@require@unique

```

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

```

1670 \RequirePackage{keyval}

```

The definitions of the keys follow.

```

1671 \define@key{ZREF@DF}{unit}{%
1672   \def\ZREF@df@unit{#1}%
1673 }
1674 \define@key{ZREF@DF}{min}{%
1675   \def\ZREF@df@min{#1}%
1676 }
1677 \define@key{ZREF@DF}{dot}{%
1678   \def\ZREF@df@dot{#1}%
1679 }

```

Defaults are set, see user interface.

```

1680 \providecommand\ZREF@df@min{2}
1681 \providecommand\ZREF@df@unit{.44em}
1682 \providecommand\ZREF@df@dot{.}

```

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

```

1683 \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.

```

1684 \newcommand*{\zdotfill}{%
1685   \leavevmode
1686   \global\advance\c@zref@unique\@ne
1687   \begingroup
1688     \def\ZREF@temp{zref@\number\c@zref@unique}%
1689     \pdfsavepos
1690     \zref@labelbyprops{\thezref@unique L}{posx}%
1691     \setlength{\dimen@}{\ZREF@df@unit}%
1692     \zref@ifrefundefined{\thezref@unique R}{%
1693       \ZREF@dotfill
1694     }{%
1695       \ifnum\numexpr\zposx{\thezref@unique R}-\zposx{\thezref@unique L}\relax
1696         <\dimexpr\ZREF@df@min\dimen@\relax
1697         \hfill
1698       \else
1699         \ZREF@dotfill
1700       \fi
1701     }%
1702     \pdfsavepos
1703     \zref@labelbyprops{\thezref@unique R}{posx}%
1704   \endgroup
1705   \kern\z@
1706 }

```

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

```

1707 \def\ZREF@dotfill{%
1708   \cleaders\hb@xt@\dimen@{\hss\ZREF@df@dot\hss}\hfill
1709 }
1710 </dotfill>

```

## 7 Test

### 7.1 `\zref@localaddprop`

```

1711 <*test1>
1712 \NeedsTeXFormat{LaTeX2e}
1713 \nofiles
1714 \documentclass{article}
1715 \usepackage{zref-base}[2010/04/17]
1716 \usepackage{qstest}
1717 \IncludeTests{*}

```

```

1718 \LogTests{log}{*}{*}
1719
1720 \makeatletter
1721 \begin{qstest}{localaddprop}{localaddprop}
1722   \Expect*{\Z@L@main}*{{default}{page}}%
1723   \zref@newprop{foobar}{F00}%
1724   \zref@newlist{alist}%
1725   \Expect*{\Z@L@alist}{}%
1726   \begingroup
1727     \zref@localaddprop{main}{foobar}%
1728     \Expect*{\Z@L@main}{{default}{page}{foobar}}%
1729     \zref@localaddprop{alist}{page}%
1730     \Expect*{\Z@L@alist}{{page}}%
1731   \endgroup
1732   \Expect*{\Z@L@main}*{{default}{page}}%
1733   \Expect*{\Z@L@alist}{}%
1734 \end{qstest}
1735 \@@end
1736 \</test1>

```

## 7.2 Module runs

```

1737 <*test-runs>
1738 \NeedsTeXFormat{LaTeX2e}
1739 \documentclass{article}
1740 \usepackage{zref-runs}[2010/04/17]
1741 \usepackage{qstest}
1742 \IncludeTests{*}
1743 \LogTests{log}{*}{*}
1744
1745 \begin{qstest}{zruns-preamble}{zruns-preamble}
1746   \Expect{0}*{\zruns}%
1747 \end{qstest}
1748
1749 \AtBeginDocument{%
1750   \begin{qstest}{zruns-atbegindocument}{zruns-atbegindocument}%
1751     \Expect*{\number\ExpectRuns}*{\zruns}%
1752   \end{qstest}%
1753 }
1754
1755 \begin{document}
1756 \begin{qstest}{zruns-document}{zruns-document}
1757   \Expect*{\number\ExpectRuns}*{\zruns}%
1758 \end{qstest}
1759 \end{document}
1760 \</test-runs>

```

## 7.3 Module titleref

```

1761 <*test-titleref-memoir>
1762 \NeedsTeXFormat{LaTeX2e}
1763 \documentclass{memoir}
1764 \usepackage{zref-titleref}[2010/04/17]
1765 \usepackage{qstest}
1766 \IncludeTests{*}
1767 \LogTests{log}{*}{*}
1768 \begin{document}
1769 \makeatletter
1770 \def\List{}
1771 \def\Label#1{%
1772   \zref@label{#1}%
1773   \g@addto@macro\List{%
1774     \par
1775     #1: [\ztitleref{#1}]%

```

```

1776 }%
1777 \mbox{}%
1778 \zref@refused{#1}%
1779 \zref@ifrefundefined{#1}{%
1780 }{%
1781   \begingroup
1782     \edef\x{\zref@extract{#1}{title}}%
1783     \Expect{OK/}*{\expandafter\ltx@carthree\x}{\}\@nil}%
1784   \endgroup
1785 }%
1786 }
1787 \def\Test#1{%
1788   \csname#1\endcsname*{OK/#1}%
1789   \Label{#1*}%
1790   \csname#1\endcsname{OK/#1}%
1791   \Label{#1}%
1792   \csname#1\endcsname[OK/#1-toc]%
1793     {WRONG-in-titleref/#1-toc-2}%
1794   \Label{#1-toc}%
1795   \expandafter\ifx\csname#1\endcsname\part
1796   \else
1797     \headnamereffalse
1798     \csname#1\endcsname[OK/#1-th-toc]%
1799       [WRONG-in-titleref/#1-th-toc-2]%
1800       {WRONG-in-titleref/#1-th-toc-3}%
1801     \Label{#1-th-toc}%
1802     \headnamereftrue
1803     \csname#1\endcsname[WRONG-in-titleref/#1-th-head-1]%
1804       [OK/#1-th-head]%
1805       {WRONG-in-titleref/#1-th-head-3}%
1806     \Label{#1-th-head}%
1807   \fi
1808 }
1809 \begin{qstest}{section}{section}
1810   \@for\x:=part,chapter,section,subsection,subsubsection\do{%
1811     \expandafter\Test\expandafter{\x}%
1812   }%
1813 \end{qstest}
1814 \newpage
1815 \List
1816 \end{document}
1817
1818 \</test-titleref-memoir>

```

## 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.

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

## 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/
```

## 8.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain `TEX`:

```
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-marks.sty</code>	→ <code>tex/latex/oberdiek/zref-marks.sty</code>
<code>zref-nextpage.sty</code>	→ <code>tex/latex/oberdiek/zref-nextpage.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-thepage.sty</code>	→ <code>tex/latex/oberdiek/zref-thepage.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>zref-example-nextpage.tex</code>	→ <code>doc/latex/oberdiek/zref-example-nextpage.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>test/zref-test-titleref-memoir.tex</code>	→ <code>doc/latex/oberdiek/test/zref-test-titleref-memoir.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 `TEX` distribution (te`TEX`, mi`TEX`, ...) relies on file name databases, you must refresh these. For example, te`TEX` 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 L<sup>A</sup>T<sub>E</sub>X.** The .dtx chooses its action depending on the format:

**plain T<sub>E</sub>X:** Run docstrip and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for docstrip (really, docstrip does not need L<sup>A</sup>T<sub>E</sub>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 Arseneau, 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 `getttitlestring`.

[2010/03/26 v2.8]

- `\zifrefundefined` added.
- Module `lastpage`: Macros `\zref@iflastpage` and `\ziflastpage` added.
- Module `thepage` added.
- Module `nextpage` added.

[2010/03/29 v2.9]

- Module `marks` added (without documentation).
- `\zref@addprop` now adds expanded property to list.
- Useless `\ZREF@ErrorNoLine` removed.

[2010/04/08 v2.10]

- Module `xr` remembers the external document name in property ‘external-document’.

[2010/04/15 v2.11]

- Module `titleref`: Better support of class `memoir`.
- Module `titleref`: Support of theorems.

[2010/04/17 v2.12]

- Module base: `\zref@newprop` ensures global empty default.
- Module xr: Setup options `tozreflabel` and `toltxlabel` added.

## 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	
<code>\@@end</code> . . . . .	1735
<code>\@addtoreset</code> . . . . .	582, 647
<code>\@auxout</code> . . . . .	450
<code>\@begintheorem</code> . . . . .	1241, 1246
<code>\@bsphack</code> . . . . .	406, 416, 1648
<code>\@caption</code> . . . . .	1103
<code>\@car</code> . . . . .	880, 1359
<code>\@cdr</code> . . . . .	882, 1360
<code>\@chapter</code> . . . . .	1115, 1146
<code>\@currentHref</code> . . . . .	1288, 1610
<code>\@currentlabel</code> . . . . .	606
<code>\@ehc</code> . . . . .	269, 283, 369, 865
<code>\@empty</code> . . . . .	263, 377, 454, 528, 610, 1038, 1042, 1068, 1075, 1220, 1307, 1336, 1357, 1398, 1548
<code>\@esphack</code> . . . . .	413, 433, 1653
<code>\@for</code> . . . . .	420, 1810
<code>\@ifclassloaded</code> . . . . .	1144, 1181
<code>\@ifdefinable</code> . . . . .	262
<code>\@ifnextchar</code> . . . . .	384, 979
<code>\@ifpackageloaded</code> . . . . .	1201, 1209, 1217, 1236
<code>\@ifstar</code> . . . . .	373, 1297
<code>\@ifundefined</code> . . . . .	192, 580, 617, 644, 662, 681, 719, 750, 838, 853, 962, 989, 1035, 1260, 1418, 1462, 1608, 1619, 1667
<code>\@input</code> . . . . .	1481
<code>\@inputcheck</code> . . . . .	1319, 1320, 1334, 1365, 1367
<code>\@latex@warning</code> . . . . .	500
<code>\@mainaux</code> . . . . .	950
<code>\@namedef</code> . . . . .	387
<code>\@ne</code> . . . . .	1002, 1686
<code>\@newl@bel</code> . . . . .	258
<code>\@nil</code> . . . . .	880, 882, 1068, 1075, 1240, 1241, 1359, 1360, 1533, 1542, 1783
<code>\@onelevel@sanitize</code> . . . . .	1239, 1242
<code>\@opargbegintheorem</code> . . . . .	1231
<code>\@part</code> . . . . .	1109
<code>\@schapter</code> . . . . .	1133
<code>\@sect</code> . . . . .	1121
<code>\@spart</code> . . . . .	1127
<code>\@ssect</code> . . . . .	1139
<code>\@testopt</code> . . . . .	1299, 1302, 1311
<code>\@tfor</code> . . . . .	293, 308, 455
<code>\@undefined</code> . . . . .	1027, 1532
<code>\\</code> . . . . .	27, 28, 29, 30, 153, 155, 157, 158, 170, 173, 1445, 1535, 1560, 1570, 1574, 1580
<code>\_</code> . . . . .	46, 47
A	
<code>\AddLineBeginAux</code> . . . . .	254
<code>\advance</code> . . . . .	686, 1002, 1686
<code>\afterassignment</code> . . . . .	758, 762
<code>\AfterLastShipout</code> . . . . .	683
<code>\Alph</code> . . . . .	7
<code>\anchor</code> . . . . .	1572
<code>\AtBeginDocument</code> . . . . .	665, 844, 944, 1101, 1749
<code>\AtBeginShipout</code> . . . . .	651, 724
<code>\AtEndOfPackage</code> . . . . .	195
B	
<code>\beamer@section</code> . . . . .	1183
<code>\beamer@subsection</code> . . . . .	1189
<code>\beamer@subsubsection</code> . . . . .	1195
<code>\begin</code> 25, 59, 102, 108, 156, 172, 1721, 1745, 1750, 1755, 1756, 1768, 1809	
<code>\bfseries</code> . . . . .	599
C	
<code>\c@abspage</code> . . . . .	654, 1008, 1013
<code>\c@page</code> . . . . .	686, 964, 1006
<code>\c@zpage</code> . . . . .	1006, 1010
<code>\c@zref@unique</code> . . . . .	586, 1002, 1686, 1688
<code>\ch@pt@c</code> . . . . .	1150, 1156, 1162
<code>\chapter</code> . . . . .	26, 32, 34, 63, 84
<code>\ChapterPages</code> . . . . .	93, 113
<code>\ChapterStart</code> . . . . .	80, 135, 150, 166
<code>\ChapterStop</code> . . . . .	87, 148, 165, 184
<code>\chardef</code> . . . . .	780, 795, 804, 808
<code>\cleaders</code> . . . . .	1708
<code>\cleardoublepage</code> . . . . .	81, 88
<code>\clearpage</code> . . . . .	64
<code>\closein</code> . . . . .	1334
<code>\comma@entry</code> . . . . .	886, 887, 888, 894, 897
<code>\comma@parse</code> . . . . .	885
<code>\csname</code> . . . . .	223, 226, 227, 263, 273, 287, 312, 326, 340, 359, 388, 389, 390, 394, 397, 400, 410, 457, 459, 464, 465, 481, 508, 519, 520, 522, 541, 542, 543, 567, 915, 923, 974, 990,



\ltx@empty	868, 1150, 1154	\openin	1319
\ltx@firstofone	228, 550, 552, 555		
\ltx@firstoftwo	274, 302, 360, 482, 531, 563, 700	<b>P</b>	
\ltx@gobble	224, 619, 620, 899, 1097	\PackageError	232, 243, 267, 281, 367, 859, 1621, 1634
\ltx@gobbletwo	582, 647, 901	\PackageInfo	264, 382, 948, 1326, 1337, 1420, 1464, 1485
\ltx@ifpackageloaded	1506	\PackageWarning	321, 336, 422, 889, 930, 1321
\ltx@ifundefined	231, 590, 729, 1145, 1147, 1620	\PackageWarningNoLine	1399, 1514, 1523
\ltx@ifundefined	1266, 1288, 1491, 1492, 1610	\page	1556
\ltx@LocalAppendToMacro	1409, 1451, 1454, 1553, 1556, 1565, 1572, 1576, 1582, 1594	\pagestyle	51
\ltx@ReturnAfterFi	1537	\par	1774
\ltx@secondoftwo	276, 304, 362, 484, 516, 529, 557, 561, 702	\part	1795
\ltx@space	801, 807, 860, 888, 931	\pdfastxpos	1643
\ltx@zero	858, 911	\pdfastypos	1644
		\pdfsavepos	1622, 1635, 1650, 1689, 1702
<b>M</b>		\pdfTeXversion	1633
\m@ne	686	\ProcessOptions	211
\M@sect	1170	\protect	499
\M@TitleReference	1500	\protected@write	450
\mainmatter	62, 134	\providecommand	255, 943, 1265, 1680, 1681, 1682
\makeatletter	11, 76, 103, 1296, 1720, 1769	\ProvidesPackage	189, 215, 614, 641, 659, 676, 716, 747, 835, 850, 941, 959, 1032, 1257, 1605, 1616, 1664
\makeatother	18, 101		
\makebox	169, 170	<b>R</b>	
\mbox	1777	\read	1365
\meaning	1241	\refstepcounter	667
\MessageBreak	244, 861, 864, 891, 893, 932, 933, 1322, 1338, 1343, 1347, 1351, 1400, 1515, 1524, 1622, 1635	\renewcommand	7, 48, 585
		\RequirePackage	191, 196, 217, 240, 245, 253, 616, 643, 645, 661, 678, 679, 680, 718, 720, 721, 749, 751, 752, 753, 754, 837, 839, 840, 852, 854, 961, 963, 1034, 1036, 1037, 1259, 1261, 1262, 1607, 1618, 1627, 1666, 1668, 1670
<b>N</b>		\reset@font	599
\NeedsTeXFormat	3, 188, 214, 613, 640, 658, 675, 715, 746, 834, 849, 940, 958, 1031, 1256, 1604, 1615, 1663, 1712, 1738, 1762	\rightarrow	47
\newcommand	20, 80, 87, 93, 167, 487, 618, 625, 634, 637, 705, 740, 757, 771, 774, 775, 841, 855, 978, 1026, 1088, 1091, 1279, 1282, 1655, 1658, 1683, 1684	\romannumeral	869
\newcounter	6, 583, 648, 969		
\newif	220, 435, 1041, 1054, 1268	<b>S</b>	
\newlabel	1436, 1447, 1480	\section	65, 137, 145
\newmarks	864	\setcounter	650
\newpage	143, 1814	\setkeys	1089, 1280, 1683
\nfss@text	599	\setlength	1691
\nofiles	1713	\SetupKeyvalOptions	1269
\NR@temp	1153, 1154	\space	501, 1339, 1340, 1344, 1348, 1352, 1622, 1635
\number	96, 111, 586, 591, 730, 945, 964, 984, 1008, 1013, 1688, 1751, 1757	\stepcounter	21, 652, 971, 972
\numexpr	96, 111, 116, 593, 732, 788, 945, 984, 1393, 1404, 1438, 1519, 1528, 1695		
		<b>T</b>	
<b>O</b>		\tableofcontents	61, 132
\on@line	947, 1336	\Test	1787, 1811
		\the	13, 153, 155, 427, 432, 463, 477, 593, 654, 692, 726, 732, 788, 1051, 1387, 1393, 1404, 1438, 1519, 1528, 1643, 1644
		\thechapter	14
		\thefoo	7, 12, 22

\theotype .....	1452	\ZREF@@@newprop .....	389, 393
\thepage .....	45, 46, 47, 448, 452, 501, 607, 1007	\ZREF@@makeperpage .....	979, 984, 988
\thezpage .....	14, 1007, 1011	\ZREF@@newprop .....	384, 386
\thezref@unique .....	9, 585, 1004, 1005, 1010, 1011, 1013, 1690, 1692, 1695, 1703	\ZREF@@perpage@step .....	993, <u>1001</u>
\title .....	1565, 1584	\zref@addprop .....	5, 15, 16, 17, 78, <u>317</u> , 608, 609, 655, 664, 723, 897, 966, 967, 968, 1040, 1612, 1645, 1646
\toks@ .....	419, 426, 427, 432, 461, 463, 476, 477, 687, 692, 1045, 1051, 1374, 1387	\ZREF@addtoks .....	475
\TR@TitleReference ..	1495, 1550, 1579	\ZREF@baseok .....	610
\ttl@sect@i .....	1203	\ZREF@call ...	780, 795, 804, 808, 816
U		\zref@default .....	7, 384, 596, <u>598</u>
\unexpanded .....	244, 249	\ZREF@df@dot .....	1678, 1682, 1708
\UniqueCounterCall .....	772	\ZREF@df@min .....	1675, 1680, 1696
\UniqueCounterNew .....	755	\ZREF@df@unit .....	1672, 1681, 1691
\url .....	1576, 1596	\ZREF@dotfill .....	1693, 1699, <u>1707</u>
\usepackage .....	9, 43, 50, 72, 74, 1715, 1716, 1740, 1741, 1764, 1765	\ZREF@extract .....	506, 511
V		\zref@extract .....	7, 97, 98, 111, 140, <u>505</u> , 632, 735, 826, 1010, 1011, 1098, 1656, 1659, 1782
\value .....	13, 726	\zref@extractdefault .	7, 117, 118, 512, <u>535</u> , 698, 699, 784, 799, 842, 1013, 1493, 1496, 1497, 1501, 1502, 1505, 1507, 1508, 1509
\verb .....	173	\ZREF@foundfalse .....	291, 1549
W		\ZREF@foundtrue .....	297, 1586
\write .....	439, 440, 950	\zref@getcurrent .....	6, <u>399</u>
X		\ZREF@gtmp .....	347, 348, 349
\x ....	293, 295, 296, 420, 421, 423, 427, 568, 570, 814, 830, 913, 918, 921, 927, 983, 986, 1238, 1239, 1244, 1373, 1376, 1380, 1384, 1394, 1397, 1407, 1408, 1439, 1444, 1782, 1783, 1810, 1811	\ZREF@iflastpage .....	706, <u>708</u> , 708
\XR@ext .....	1266	\zref@iflastpage .....	10, <u>697</u> , 711
Y		\ZREF@iflistcontainsprop ...	287, 289
\y .....	292, 296, 1240, 1241	\zref@iflistcontainsprop .....	5, <u>286</u> , 320, 335
Z		\zref@iflistundefined .	5, 261, <u>272</u> , 280
\z .....	1241, 1242, 1244	\zref@ifpropundefined .....	6, <u>358</u> , 366, 421, 910, 1283, 1286, 1291, 1448, 1543
\z@ .....	979, 1705	\ZREF@ifrefcontainsprop ...	518, 526
\Z@D@page .....	775	\zref@ifrefcontainsprop .....	7, <u>514</u> , 1590, 1591
\Z@L@alist .....	1725, 1730, 1733	\ZREF@ifrefundefined .....	488, <u>490</u> , 781, 792, 802
\Z@L@LastPage .....	689	\zref@ifrefundefined .	7, <u>480</u> , 492, 498, 515, 536, 793, 1005, 1692, 1779
\Z@L@main .....	688, 1722, 1728, 1732	\ZREF@immediatetrue .....	438
\zdotfill .....	15, 170, 173, <u>1684</u>	\ZREF@l@addto@macro .....	340, <u>345</u>
\zdotfillsetup .....	15, <u>1683</u>	\ZREF@label .....	408, 432, <u>444</u> , 692
\zexternaldocument .....	16, <u>1282</u>	\zref@label .....	6, <u>402</u> , 622, 1772
\ziflastpage .....	10, <u>705</u>	\zref@labelbylist .....	6, 403, <u>405</u> , 726, 1004, 1651
\zifrefundefined .....	7, <u>487</u>	\zref@labelbyprops .....	6, 90, <u>415</u> , 779, 1690, 1703
\zlabel .....	9, 85, 106, 138, 146, <u>618</u>	\zref@listexists .	5, <u>279</u> , 318, 333, 407
\zmakeperpage .....	13, <u>978</u>	\zref@listforloop .....	<u>307</u>
\znextpage .....	12, 53, 56, 771	\zref@localaddprop .	5, <u>332</u> , 1727, 1729
\znextpagesetup .....	12, 44, 757	\ZREF@mainlist .....	403, 602, 605, 608, 609, 655, 664, 1040, 1612
\znonextpagename .....	48, 774, 822	\ZREF@makeperpage@opt .....	979, 981
\zpageref .....	9, 127, <u>634</u>	\ZREF@MARKS@DefineProp .....	873, 874, 875, 909
\zposx .....	15, 153, <u>1655</u> , 1695	\zref@marks@register	855, 860, 892, 931
\zposy .....	15, 155, <u>1655</u>		
\zref .....	9, 27, 28, 29, 30, 113, 115, 124, 129, 139, <u>625</u> , 635		

\ZREF@name	218,	\zref@refused	7, 491, 494, 631,
232, 243, 267, 281, 321, 336,		637, 709, 710, 738, 845, 1096, 1778	
367, 422, 859, 889, 930, 1621, 1634		\zref@require@unique	9, 579, 970, 1669
\ZREF@NAME@bot	888, 908	\zref@setcurrent	6, 83, 391, 396, 668
\ZREF@NAME@first	887, 907	\zref@setdefault	7, 595, 598
\ZREF@NAME@top	886, 906	\zref@setmainlist	8, 601
\zref@newlabel		\ZREF@STAR	881, 905
6, 255, 257, 470, 1391, 1479		\ZREF@stripperperiod	1067, 1075
\zref@newlist	5, 260,	\ZREF@temp	193, 200,
605, 682, 722, 883, 965, 1642, 1724		201, 202, 203, 204, 205, 206,	
\ZREF@newprop	375, 378, 381	207, 208, 209, 210, 325, 327,	
\zref@newprop	6,	454, 461, 462, 470, 880, 881, 1688	
12, 13, 14, 77, 372, 606, 607,		\ZREF@TempName	856, 868,
654, 663, 914, 922, 964, 1039,		869, 871, 897, 910, 914, 922, 933	
1263, 1264, 1284, 1287, 1292,		\ZREF@TempNum	
1449, 1544, 1609, 1643, 1644, 1723		857, 858, 862, 869, 911, 924	
\ZREF@nextpage	772, 776	\zref@thepage	11, 734, 742
\ZREF@nil	394, 528, 543, 1366, 1372,	\zref@thepage@name	
1377, 1381, 1391, 1407, 1436,		11, 729, 735, 738, 787	
1444, 1531, 1538, 1547, 1550, 1579		\zref@thepage@refused	737, 741
\ZREF@NOVALUE	534	\ZREF@titleref	1092, 1094
\ZREF@novalue	527, 528, 534	\zref@titleref@cleanup	1043, 1083
\ZREF@np@call@next	766, 770, 825	\zref@titleref@current	
\ZREF@np@call@nonext	763, 769, 821	1038, 1039, 1062, 1066, 1067, 1086	
\ZREF@np@call@unknown	759, 768, 817	\ZREF@titleref@hook	
\ZREF@np@setup@i	758, 761	1042, 1046, 1050, 1073	
\ZREF@np@setup@ii	762, 765	\zref@titleref@setcurrent	
\ZREF@number	590, 857	1056, 1104,	
\ZREF@org@begintheorem	1248	1110, 1116, 1122, 1128, 1134,	
\ZREF@org@caption	1105	1140, 1148, 1151, 1155, 1159,	
\ZREF@org@chapter	1117, 1167	1161, 1172, 1174, 1184, 1190,	
\ZREF@org@opargbegintheorem	1233	1196, 1204, 1213, 1222, 1232, 1247	
\ZREF@org@part	1111	\zref@titleref@stripperperiodtrue	1055
\ZREF@org@schapter	1135	\ZREF@unexpanded	537, 539, 548, 550, 552
\ZREF@org@ssect	1123	\ZREF@UpdatePdfTeX	219, 1624, 1637
\ZREF@org@spart	1129	\ZREF@wrapper@babel	570, 576
\ZREF@org@ssect	1141	\zref@wrapper@babel	8, 140,
\ZREF@org@beamer@section	1185	488, 495, 553, 622, 627, 706, 1092	
\ZREF@org@beamer@subsection	1191	\zref@wrapper@immediate	
\ZREF@org@beamer@subsubsection	1197	8, 89, 436, 691, 725	
\ZREF@org@lst@MakeCaption	1226	\zref@wrapper@unexpanded	8, 547
\ZREF@org@LT@caption	1212	\ZREF@X	374, 377, 388
\ZREF@org@M@ssect	1176	\zref@x@	1277
\ZREF@org@refstepcounter	669	\ZREF@xr@input	1384, 1481
\ZREF@org@stepcounter	971, 976	\ZREF@xr@checkfile	1315, 1318, 1361
\ZREF@org@thepage	448, 452	\ZREF@xr@checkkey	1533, 1542
\ZREF@org@ttl@ssect@i	1205	\ZREF@xr@checklist	1407, 1531
\ZREF@org@write	439, 440	\zref@xr@ext	16, 1265, 1311
\ZREF@P	383, 387, 388, 389, 390,	\ZREF@xr@externaldocument	
391, 394, 455, 457, 459, 464, 465		1299, 1302, 1305	
\ZREF@pagenum@last	798, 801	\ZREF@xr@externalfile	
\ZREF@pagenum@this		1308, 1309, 1413, 1457	
783, 788, 791, 801, 807		\ZREF@xr@file	1309, 1319, 1322,
\ZREF@patch	221, 666,	1327, 1338, 1359, 1401, 1516, 1525	
1102, 1108, 1114, 1120, 1126,		\ZREF@xr@filelist	
1132, 1138, 1169, 1182, 1188,		1307, 1357, 1359, 1360, 1385, 1386	
1194, 1202, 1210, 1218, 1230, 1245		\ZREF@xr@found	1329, 1339, 1393, 1438
\zref@prop	309, 314	\ZREF@xr@graburl	1311, 1313
\zref@propexists	6, 319, 334, 365, 626	\ZREF@xr@ignored	1519
\ZREF@refname@next	786, 793, 802, 826	\ZREF@xr@ignored@empty	
\ZREF@refname@this	778, 779, 781, 784	1330, 1342, 1344, 1403, 1404, 1518	
\ZREF@refused	495, 497		

\ZREF@xr@ignored@ltx .....	\ZREF@xr@temprefname .....
..... 1332, 1350, 1352, 1527, 1528	..... 1396, 1408, 1410,
\ZREF@xr@ignored@zref 1331, 1346, 1348	1426, 1441, 1443, 1452, 1455, 1470
\ZREF@xr@line .. 1365, 1366, 1377, 1381	\ZREF@xr@tolabel ... 1432, 1476, 1483
\ZREF@xr@list .....	\ZREF@xr@url .....
1397, 1398	1314, 1596
\ZREF@xr@ltx@ignorewarning .... 1522	\ZREF@xr@urlcheck .. 1416, 1460, 1589
\ZREF@xr@newlabel .....	\ZREF@xr@zref@ignorewarning ....
1380, 1480	..... 1428, 1472, 1513
\ZREF@xr@prefix . 1306, 1392, 1428,	\ZREF@xr@zref@newlabel .. 1376, 1479
1432, 1437, 1460, 1465, 1472, 1476	\ZREF@xr@zreflabelfalse .....
\ZREF@xr@process@label .. 1381, 1436	1298
\ZREF@xr@process@zreflabel 1377, 1391	\ZREF@xr@zreflabeltrue .....
\ZREF@xr@processfile .... 1318, 1364	1301
\ZREF@xr@processline .... 1366, 1372	\ZREF@zref .....
\ZREF@xr@refname .....	627, 630
1392, 1418, 1425, 1437, 1462, 1469	\zrefused . 9, 94, 95, 161, 162, 163, 637
\ZREF@xr@relax .....	\zruns .....
1482, 1563	13, 943, 1746, 1751, 1757
\ZREF@xr@scanparams .....	\zsavepos .....
1442, 1547	15, 157, 158, 1647
\ZREF@xr@scantitleref ... 1550, 1579	\zthepage .....
\ZREF@xr@temp .....	11, 740
1562, 1563	\ztitleref .....
\ZREF@xr@tempname ... 1395, 1396,	14, 1091, 1775
1416, 1421, 1432, 1440, 1441, 1476	\ztitlerefsetup .....
	14, 1076
	\ztotpages .....
	13, 125, 841
	\zunknownnextpagename .. 12, 775, 818
	\zunmakeperpage .....
	14, 1026
	\zxrsetup .....
	16, 1279