

# The zref package

Heiko Oberdiek

<heiko.oberdiek at gmail.com>

2010/03/26 v2.8

## 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	4
2.2	Property list	5
2.3	Property	5
2.4	Reference generation	6
2.5	Data extraction	6
2.6	Setup	7
2.7	Declared properties	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>	9
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>runs</code>	13
3.8	Module <code>perpage</code>	13
3.9	Module <code>counter</code>	14
3.10	Module <code>titleref</code>	14
3.11	Module <code>savepos</code>	15
3.12	Module <code>dotfill</code>	15
3.13	Module <code>xr</code>	16
<b>4</b>	<b>ToDo</b>	<b>16</b>

<b>5</b>	<b>Example</b>	<b>16</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>	19
6.2.1	Prefixes	19
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	Setup	29
6.3	Module <code>user</code>	29
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>runs</code>	34
6.11	Module <code>perpage</code>	35
6.12	Module <code>titleref</code>	37
6.12.1	Implementation	37
6.12.2	User interface	38
6.12.3	Patches for section and caption commands	39
6.13	Module <code>xr</code>	40
6.14	Module <code>hyperref</code>	45
6.15	Module <code>savepos</code>	45
6.15.1	Identification	45
6.15.2	Availability	45
6.15.3	Setup	46
6.15.4	User macros	46
6.16	Module <code>dotfill</code>	47
<b>7</b>	<b>Test</b>	<b>48</b>
7.1	<code>\zref@localaddprop</code>	48
7.2	Module <code>runs</code>	48
<b>8</b>	<b>Installation</b>	<b>49</b>
8.1	Download	49
8.2	Bundle installation	49
8.3	Package installation	49
8.4	Refresh file name databases	50
8.5	Some details for the interested	50
<b>9</b>	<b>References</b>	<b>51</b>

<b>10 History</b>	<b>51</b>
[2006/02/20 v1.0]	51
[2006/05/03 v1.1]	51
[2006/05/25 v1.2]	51
[2006/09/08 v1.3]	51
[2007/01/23 v1.4]	51
[2007/02/18 v1.5]	52
[2007/04/06 v1.6]	52
[2007/04/17 v1.7]	52
[2007/04/22 v1.8]	52
[2007/05/02 v1.9]	52
[2007/05/06 v2.0]	52
[2007/05/28 v2.1]	52
[2008/09/21 v2.2]	52
[2008/10/01 v2.3]	52
[2009/08/07 v2.4]	52
[2009/12/06 v2.5]	52
[2009/12/07 v2.6]	53
[2009/12/08 v2.7]	53
[2010/03/26 v2.8]	53
<b>11 Index</b>	<b>53</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 appearance 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 L<sup>A</sup>T<sub>E</sub>X kernel. An interface for adding new properties is missing.

There are several tries to add new properties:

**hyperref** uses a list of five properties instead of the standard list with two entries.

This causes many compatibility problems with L<sup>A</sup>T<sub>E</sub>X and other packages.

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

## 1.2 Basic idea

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

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

The entries have the following format:

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

This approach is much more flexible:

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

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

## 1.3 Interfaces

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

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

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

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

```
\RequirePackage{zref-abspage}
```

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

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

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

## 2 Interface for programmers

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

### 2.1 Entities

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

```
\Z@R@myref → \default{2.1}\page{7}
```

The generic format of a entry is:

```
\Z@R@(\refname) → \(\propname){\langle value \rangle}
```

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

## 2.2 Property list

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

`\zref@newlist {<listname>}`

Declares a new empty property list.

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

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

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

Local variant of `\zref@addprop`.

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

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

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

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

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

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

## 2.3 Property

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

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

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

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

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

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

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

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

`\zref@propexists {<propname>} {<then>}`

Calls `<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:

```

\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 `\ref@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 {⟨value⟩}`

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

## 2.7 Declared properties

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

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

`\zlabel {⟨refname⟩}^{babel}`

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

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

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

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

`\zpageref {⟨refname⟩}^{babel}`

Convenience macro, similar to `\pageref`.

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

`\zrefused {⟨refname⟩}^{babel}`

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

### 3.2 Module `abspage`

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

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

The example also makes use of module `lastpage`.

### 3.3 Module `lastpage`

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

`\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}[2008/10/01]
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 \end{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}[2010/03/26]
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]{\znexpage}%
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 \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 `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`).

### 3.8 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}
```

<code>\thezpage</code> <code>counter zpage</code>
--

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}}
```

<code>\zunmakeperpage {⟨counter⟩}</code>
--

The reset mechanism for this counter is deactivated.

### 3.9 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 reimplementaion of the `autoref` feature, see the section 4 with the todo list.

### 3.10 Module titleref

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

<code>\ztitleref {⟨refname⟩}^babel</code>
---

Print the section or caption title of reference `⟨refname⟩`, similar to `\nameref` or `\titleref`.

<code>\ztitlerefsetup {key<sub>1</sub>=value<sub>1</sub>, key<sub>2</sub>=value<sub>2</sub>, ...}</code>
--

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.

### 3.11 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.

### 3.12 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`).

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

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

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

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

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

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

`\zref@xr@ext`

If the `<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 \pageref{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   \zsavapos{firstcol}Hello&\zsavapos{secondcol}World\\
158   \zsavapos{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
166
Test for module \dotfill.
166 \ChapterStart{dotfill}{Test for dotfill feature}
167 \newcommand*{\dfptest}[1]{%
168   #1&
169   [\makebox[{#1}]{\dotfill}]&
170   [\makebox[{#1}]{\zdotfill}]\,
171 }
172 \begin{tabular}{rll}
173 & [\verb|\dotfill|] & [\verb|\zdotfill|]\,
174 \dfptest{0.43em}
175 \dfptest{0.44em}
176 \dfptest{0.45em}
177 \dfptest{0.87em}
178 \dfptest{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 [2010/03/26 v2.8 New reference scheme for LaTeX2e (H0)]%

```

#### 6.1.2 Load basic module

```

191 \RequirePackage{zref-base}[2010/03/26]

```

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}[2010/03/26]%
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 *<listname>*.

**\Z@D@propname:** The default value for property *<propname>*.

**\Z@E@propname:** Extract function for property *<propname>*.

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

`\Z@C@propname`: Current value of the property `<propname>`.

`\Z@R@labelname`: Data for reference `<labelname>`.

`\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 [2010/03/26 v2.8 Module base for zref (H0)]%
```

### 6.2.3 Utilities

```
217 \RequirePackage{ltxcmds}[2009/12/12]
```

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

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

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

```
219 \def\ZREF@ErrorNoLine#1#2{%
220   \begingroup
221     \let\on@line\@empty
222     \PackageError\ZREF@name{#1}{#2}%
223   \endgroup
224 }
```

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

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

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

```
226 \newif\ifZREF@found
```

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

```
227 \def\ZREF@patch#1{%
228   \begingroup\expandafter\expandafter\expandafter\endgroup
229   \expandafter\ifx\csname #1\endcsname\relax
230     \expandafter\ltx@gobble
231   \else
232     \expandafter\let\csname ZREF@org@#1\expandafter\endcsname
233     \csname #1\endcsname
234     \expandafter\ltx@firstofone
235   \fi
236 }
```

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

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

```

237 \ltx@ifundefined{eTeXversion}{%
238   \ZREF@ErrorNoLine{%
239     Missing support for eTeX; package is abandoned%
240   }{%
241     Use a TeX compiler that support eTeX and enable eTeX %
242     in the format.%
243   }%
244   \endinput
245 }{}%

246 \RequirePackage{etexcmds}[2007/09/09]
247 \ifetex@unexpanded
248 \else
249   \ZREF@ErrorNoLine{%
250     Missing e-TeX's \string\unexpanded.\MessageBreak
251     Add \string\RequirePackage\string{etexcmds\string} before %
252     \string\documentclass%
253   }{%
254     Probably you are using some package (e.g. ConTeXt) that %
255     redefines \string\unexpanded%
256   }%
257   \expandafter\endinput
258 \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.

```

259 \RequirePackage{auxhook}
260 \AddLineBeginAux{%
261   \string\providecommand\string\zref@newlabel[2]{}%
262 }

```

`\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.

```

263 \def\zref@newlabel{%
264   \@newl@bel{Z@R}%
265 }

```

### 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.

```

266 \def\zref@newlist#1{%
267   \zref@iflistundefined{#1}{%
268     \ifdefinable{Z@L@#1}{%
269       \global\expandafter\let\csname Z@L@#1\endcsname\@empty
270       \PackageInfo{zref}{New property list: #1}%
271     }%
272   }{%

```

```

273     \PackageError\ZREF@name{%
274         Property list '#1' already exists%
275     }\@ehc
276 }%
277 }

```

`\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.

```

278 \def\zref@iflistundefined#1{%
279     \expandafter\ifx\csname Z@L@#1\endcsname\relax
280     \expandafter\ltx@firstoftwo
281 }else
282     \expandafter\ltx@secondoftwo
283 \fi
284 }

```

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

```

285 \def\zref@listexists#1{%
286     \zref@iflistundefined{#1}{%
287         \PackageError\ZREF@name{%
288             Property list '#1' does not exist%
289         }\@ehc
290     }%
291 }

```

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

```

292 \def\zref@iflistcontainsprop#1{%
293     \expandafter\ZREF@iflistcontainsprop\csname Z@L@#1\endcsname
294 }
295 \def\ZREF@iflistcontainsprop#1#2{%
296     \begingroup
297     \ZREF@foundfalse
298     \edef\y{#2}%
299     \expandafter\@tfor\expandafter\x
300     \expandafter:\expandafter=#1\do{%
301         \edef\x{x}%
302         \ifx\x\y
303             \ZREF@foundtrue
304         \fi
305     }%
306     \expandafter\endgroup
307     \ifZREF@found
308         \expandafter\ltx@firstoftwo
309     \else
310         \expandafter\ltx@secondoftwo
311     \fi
312 }

```

`\zref@listforloop`

```

313 \def\zref@listforloop#1#2{%
314     \expandafter\expandafter\expandafter\@tfor
315     \expandafter\expandafter\expandafter\zref@prop
316     \expandafter\expandafter\expandafter:%
317     \expandafter\expandafter\expandafter=%
318     \csname Z@L@#1\endcsname
319     \do{%
320         #2\zref@prop
321     }%
322 }

```

`\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.

```

323 \def\zref@addprop#1#2{%
324   \zref@listexists{#1}{%
325     \zref@propexists{#2}{%
326       \zref@iflistcontainsprop{#1}{#2}{%
327         \PackageWarning\ZREF@name{%
328           Property ‘#2’ is already in list ‘#1’%
329         }%
330       }%
331       \expandafter\g@addto@macro\csname Z@L@#1\endcsname{{#2}}%
332     }%
333   }%
334 }%
335 }
```

`\zref@localaddprop`

```

336 \def\zref@localaddprop#1#2{%
337   \zref@listexists{#1}{%
338     \zref@propexists{#2}{%
339       \zref@iflistcontainsprop{#1}{#2}{%
340         \PackageWarning\ZREF@name{%
341           Property ‘#2’ is already in list ‘#1’%
342         }%
343       }%
344       \expandafter\ZREF@l@addto@macro\csname Z@L@#1\endcsname{{#2}}%
345     }%
346   }%
347 }%
348 }
```

`\ZREF@l@addto@macro`

```

349 \ifetex@unexpanded
350   \def\ZREF@l@addto@macro#1#2{%
351     \global\let\ZREF@gtemp#1%
352     \g@addto@macro\ZREF@gtemp{#2}%
353     \let#1\ZREF@gtemp
354   }%
355 \else
356   \def\ZREF@l@addto@macro#1#2{%
357     \edef#1{%
358       \etex@unexpanded\expandafter{#1#2}%
359     }%
360   }%
361 \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.

```

362 \def\zref@ifpropundefined#1{%
363   \expandafter\ifx\csname Z@E@#1\endcsname\relax
364     \expandafter\ltx@firstoftwo
365   \else
366     \expandafter\ltx@secondoftwo
367   \fi
368 }
```

`\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.

```

369 \def\zref@propexists#1{%
```

```

370 \zref@ifpropundefined{#1}{%
371   \PackageError\ZREF@name{%
372     Property ‘#1’ does not exist%
373   }\@ehc
374 }%
375 }

```

**\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.

```

376 \def\zref@newprop{%
377   \ifstar{%
378     \let\ZREF@X\noexpand
379     \ZREF@newprop
380   }{%
381     \let\ZREF@X\@empty
382     \ZREF@newprop
383   }%
384 }
385 \def\ZREF@newprop#1{%
386   \PackageInfo{zref}{New property: #1}%
387   \def\ZREF@P{#1}%
388   \ifnextchar[\ZREF@@newprop{\ZREF@@newprop[\zref@default]}%
389 }
390 \def\ZREF@@newprop[#1]{%
391   \global\@namedef{Z@D@\ZREF@P}{#1}%
392   \global\expandafter\let\csname Z@X@\ZREF@P\endcsname\ZREF@X
393   \expandafter\ZREF@@@newprop\csname\ZREF@P\endcsname
394   \zref@setcurrent\ZREF@P
395 }
396 \def\ZREF@@@newprop#1{%
397   \expandafter\gdef\csname Z@E@\ZREF@P\endcsname##1##2##3\ZREF@nil{##2}%
398 }

```

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

```

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

```

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

```

402 \def\zref@getcurrent#1{%
403   \csname Z@C@#1\endcsname
404 }

```

## 6.2.8 Reference generation

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

```

405 \def\zref@label#1{%
406   \zref@labelbylist{#1}\ZREF@mainlist
407 }

```



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

```

408 \def\zref@labelbylist#1#2{%
409   \@bsphack
410   \zref@listexists{#2}{%
411     \expandafter\expandafter\expandafter\ZREF@label
412     \expandafter\expandafter\expandafter{%
413       \csname Z@L@#2\endcsname
414     }{#1}%
415   }%
416   \@esphack
417 }

```

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

```

418 \def\zref@labelbyprops#1#2{%
419   \@bsphack
420   \begingroup
421   \edef\l{#2}%
422   \toks@{}%
423   \@for\x:=#2\do{%
424     \zref@ifpropundefined{\x}{%
425       \PackageWarning\ZREF@name{%
426         Property ‘\x’ is not known%
427       }%
428     }{%
429       \toks@\expandafter\expandafter\expandafter{%
430         \expandafter\the\expandafter\toks@\expandafter{\x}%
431       }%
432     }%
433   }%
434   \expandafter\endgroup
435   \expandafter\ZREF@label\expandafter{\the\toks@}{#1}%
436   \@esphack
437 }

```

`\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.

```

438 \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`.

```

439 \long\def\zref@wrapper@immediate#1{%
440   \begingroup
441   \ZREF@immediatetrue
442   \let\ZREF@org@write\write
443   \def\write{\immediate\ZREF@org@write}%
444   #1%
445   \endgroup
446 }

```

`\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.

```

447 \def\ZREF@label#1#2{%
448   \if@filesw
449     \begingroup
450     \ifZREF@immediate
451       \let\ZREF@org@thepage\thepage
452     \fi

```

```

453 \protected@write\@auxout{%
454 \ifZREF@immediate
455 \let\thepage\ZREF@org@thepage
456 \fi
457 \let\ZREF@temp\@empty
458 \@tfor\ZREF@P:=#1\do{%
459 \expandafter\ifx
460 \csname\ifZREF@immediate relax\else Z@X@\ZREF@P\fi\endcsname
461 \noexpand
462 \expandafter\let\csname Z@C@\ZREF@P\endcsname\relax
463 \fi
464 \toks@\expandafter{\ZREF@temp}%
465 \edef\ZREF@temp{%
466 \the\toks@
467 \expandafter\string\csname\ZREF@P\endcsname{%
468 \expandafter\noexpand\csname Z@C@\ZREF@P\endcsname
469 }%
470 }%
471 }%
472 }{%
473 \string\zref@newlabel{#2}{\ZREF@temp}%
474 }%
475 \endgroup
476 \fi
477 }
478 \def\ZREF@addtoks#1{%
479 \toks@\expandafter\expandafter\expandafter{%
480 \expandafter\the\expandafter\toks@#1%
481 }%
482 }

```

### 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.

```

483 \def\zref@ifrefundefined#1{%
484 \expandafter\ifx\csname Z@R@#1\endcsname\relax
485 \expandafter\ltx@firstoftwo
486 \else
487 \expandafter\ltx@secondoftwo
488 \fi
489 }

```

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

```

490 \newcommand*\zifrefundefined}[1]{%
491 \zref@wrapper@babel\ZREF@ifrefundefined{#1}%
492 }

```

`\ZREF@ifrefundefined`

```
493 \def\ZREF@ifrefundefined#1{%
494   \zref@refused{#1}%
495   \zref@ifrefundefined{#1}%
496 }
```

`\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.

```
497 \def\zref@refused#1{%
498   \zref@wrapper@babel\ZREF@refused{#1}%
499 }
```

`\ZREF@refused`

```
500 \def\ZREF@refused#1{%
501   \zref@ifrefundefined{#1}{%
502     \protect\G@refundefinedtrue
503     \@latex@warning{%
504       Reference ‘#1’ on page \thepage \space undefined%
505     }%
506   }{}%
507 }
```

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

```
508 \def\zref@extract#1#2{%
509   \expandafter\expandafter\expandafter\ZREF@extract
510   \expandafter\expandafter\expandafter{%
511     \csname Z@D@#2\endcsname
512   }{#1}{#2}%
513 }
514 \def\ZREF@extract#1#2#3{%
515   \zref@extractdefault{#2}{#3}{#1}%
516 }
```

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

```
517 \def\zref@ifrefcontainsprop#1#2{%
518   \zref@ifrefundefined{#1}{%
519     \ltx@secondoftwo
520   }{%
521     \expandafter\ZREF@ifrefcontainsprop
522     \csname Z@E@#2\expandafter\endcsname
523     \csname#2\expandafter\expandafter\expandafter\endcsname
524     \expandafter\expandafter\expandafter{%
525       \csname Z@R@#1\endcsname
526     }%
527   }%
528 }
529 \def\ZREF@ifrefcontainsprop#1#2#3{%
530   \expandafter\ifx\expandafter\ZREF@novalue
531   #1#3#2\ZREF@novalue\ZREF@nil\@empty
532   \expandafter\ltx@secondoftwo
533   \else
534   \expandafter\ltx@firstoftwo
535   \fi
536 }
537 \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.

```

538 \def\zref@extractdefault#1#2#3{%
539   \zref@ifrefundefined{#1}{%
540     \ZREF@unexpanded{#3}%
541   }{%
542     \expandafter\expandafter\expandafter\ZREF@unexpanded
543     \expandafter\expandafter\expandafter{%
544       \csname Z@E@#2\expandafter\expandafter\expandafter\endcsname
545       \csname Z@R@#1\expandafter\endcsname
546       \csname#2\endcsname{#3}\ZREF@nil
547     }%
548   }%
549 }

```

`\zref@wrapper@unexpanded`

```

550 \long\def\zref@wrapper@unexpanded#1{%
551   \let\ZREF@unexpanded\etex@unexpanded
552   #1%
553   \let\ZREF@unexpanded\ltx@firstofone
554 }
555 \let\ZREF@unexpanded\ltx@firstofone

```

## 6.2.10 Compatibility with babel

`\zref@wrapper@babel`

```

556 \long\def\zref@wrapper@babel#1#2{%
557   \ifcsname if@safe@actives\endcsname
558     \expandafter\ltx@firstofone
559   \else
560     \expandafter\ltx@secondoftwo
561   \fi
562   {%
563     \if@safe@actives
564       \expandafter\ltx@secondoftwo
565     \else
566       \expandafter\ltx@firstoftwo
567     \fi
568     {%
569       \begingroup
570       \csname @safe@activestrue\endcsname
571       \edef\x{#2}%
572       \expandafter\endgroup
573       \expandafter\ZREF@wrapper@babel\expandafter{\x}{#1}%
574     }%
575   }{%
576     #1{#2}%
577   }%
578 }
579 \long\def\ZREF@wrapper@babel#1#2{%
580   #2{#1}%
581 }

```

## 6.2.11 Unique counter support

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

```

582 \def\zref@require@unique{%
583   \@ifundefined{c@zref@unique}{%
584     \begingroup
585     \let\@addtoreset\ltx@gobbletwo
586     \newcounter{zref@unique}%
587     \endgroup

```

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

```

588 \renewcommand*{\thezref@unique}{%
589   zref@number\c@zref@unique
590 }%
591 }{}%
592 }

```

### 6.2.12 Setup

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

```

593 \def\zref@setdefault#1{%
594   \def\zref@default{#1}%
595 }

```

`\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.

```

596 \zref@setdefault{%
597   \nfss@text{\reset@font\bfseries ??}%
598 }

```

#### Main property list.

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

```

599 \def\zref@setmainlist#1{%
600   \def\ZREF@mainlist{#1}%
601 }
602 \zref@setmainlist{main}

```

Now we create the list.

```

603 \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

```

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

```

#### Mark successful loading

```

608 \let\ZREF@baseok\@empty
609 \</base>

```

## 6.3 Module user

```

610 <*user>
611 \NeedsTeXFormat{LaTeX2e}
612 \ProvidesPackage{zref-user}%
613 [2010/03/26 v2.8 Module user for zref (HO)]%
614 \RequirePackage{zref-base}[2010/03/26]
615 \@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.

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

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

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

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

```
632 \newcommand*\zpageref{%
633   \zref[page]%
634 }%
```

`\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.

```
635 \newcommand*\zrefused{\zref@refused}%
636 </user>
```

## 6.4 Module `abspage`

```
637 <*abspage>
638 \NeedsTeXFormat{LaTeX2e}
639 \ProvidesPackage{zref-abspage}%
640 [2010/03/26 v2.8 Module abspage for zref (HO)]%
641 \RequirePackage{zref-base}[2010/03/26]
642 \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`.

```
643 \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.

```
644 \begingroup
645   \let\@addtoreset\ltx@gobbletwo
646   \newcounter{abspage}%
647 \endgroup
648 \setcounter{abspage}{0}%
649 \AtBeginShipout{%
650   \stepcounter{abspage}%
651 }%
652 \zref@newprop*{abspage}[0]{\the\c@abspage}%
```

```
653 \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.

```
654 \abspage
```

## 6.5 Module counter

```
655 \*counter
656 \NeedsTeXFormat{LaTeX2e}
657 \ProvidesPackage{zref-counter}%
658 [2010/03/26 v2.8 Module counter for zref (H0)]%
659 \RequirePackage{zref-base}[2010/03/26]
660 \@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.

```
661 \zref@newprop{counter}{}
662 \zref@addprop\ZREF@mainlist{counter}
```

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

```
663 \AtBeginDocument{%
664   \ZREF@patch{refstepcounter}{%
665     \def\refstepcounter#1{%
666       \zref@setcurrent{counter}{#1}%
667       \ZREF@org@refstepcounter{#1}%
668     }%
669   }%
670 }
671 \counter
```

## 6.6 Module lastpage

```
672 \*lastpage
673 \NeedsTeXFormat{LaTeX2e}
674 \ProvidesPackage{zref-lastpage}%
675 [2010/03/26 v2.8 Module lastpage for zref (H0)]%
676 \RequirePackage{zref-base}[2010/03/26]
677 \RequirePackage{zref-abspage}[2010/03/26]
678 \RequirePackage{atveryend}[2009/12/07]
679 \@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.

```
680 \zref@newlist{LastPage}
681 \AfterLastShipout{%
682   \if@filesw
683     \begingroup
684       \advance\c@page\m@ne
685       \toks@\expandafter\expandafter\expandafter{%
686         \expandafter\Z@L@main
687         \Z@L@LastPage
688       }%
689       \expandafter\zref@wrapper@immediate\expandafter{%
690         \expandafter\ZREF@label\expandafter{\the\toks@}{LastPage}%
691       }%
692     \endgroup
693   \fi
694 }
```

```
\zref@iflastpage
```

```

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

\ziflastpage
703 \newcommand*{\ziflastpage}{%
704   \zref@wrapper@babel\ZREF@iflastpage
705 }

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

711 </lastpage>

```

## 6.7 Module thepage

```

712 <*thepage>
713 \NeedsTeXFormat{LaTeX2e}
714 \ProvidesPackage{zref-thepage}%
715   [2010/03/26 v2.8 Module thepage for zref (H0)]%
716 \RequirePackage{zref-base}[2010/03/26]
717 \@ifundefined{ZREF@baseok}{\endinput}{}

718 \RequirePackage{atbegshi}
719 \RequirePackage{zref-abspage}[2010/03/26]

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

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

\zref@thepage
732 \def\zref@thepage#1{%
733   \zref@extract{\zref@thepage@name{#1}}{page}%
734 }%

\zref@thepage@refused
735 \def\zref@thepage@refused#1{%
736   \zref@refused{\zref@thepage@name{#1}}%
737 }%

\zthepage
738 \newcommand*{\zthepage}[1]{%
739   \zref@thepage@refused{#1}%

```



```

740 \zref@thepage{#1}%
741 }

742 </thepage>

```

## 6.8 Module nextpage

```

743 <*nextpage>
744 \NeedsTeXFormat{LaTeX2e}
745 \ProvidesPackage{zref-nextpage}%
746 [2010/03/26 v2.8 Module nextpage for zref (HO)]%
747 \RequirePackage{zref-base}[2010/03/26]
748 \@ifundefined{ZREF@baseok}{\endinput}{}

749 \RequirePackage{zref-abspage}[2010/03/26]
750 \RequirePackage{zref-thepage}[2010/03/26]
751 \RequirePackage{zref-lastpage}[2010/03/26]
752 \RequirePackage{uniquecounter}[2009/12/18]

753 \UniqueCounterNew{znnextpage}
754
755 \newcommand*{\znnextpagesetup}{%
756 \afterassignment\ZREF@np@setup@i
757 \def\ZREF@np@call@unknown##1%
758 }
759 \def\ZREF@np@setup@i{%
760 \afterassignment\ZREF@np@setup@ii
761 \def\ZREF@np@call@nonext##1%
762 }
763 \def\ZREF@np@setup@ii{%
764 \def\ZREF@np@call@next##1%
765 }
766 \def\ZREF@np@call@unknown#1{#1}
767 \def\ZREF@np@call@nonext#1{#1}
768 \def\ZREF@np@call@next#1{#1}
769 \newcommand*{\znnextpage}{%
770 \UniqueCounterCall{znnextpage}{\ZREF@nextpage}%
771 }
772 \newcommand*{\znonextpagename}{}
773 \newcommand*{\zunknownnextpagename}{\Z@D@page}
774 \def\ZREF@nextpage#1{%
775 \begingroup
776 \def\ZREF@refname@this{zref@np#1}%
777 \zref@labelbyprops\ZREF@refname@this{abspage}%
778 \chardef\ZREF@call=0 % unknown
779 \ZREF@ifrefundefined\ZREF@refname@this{%
780 }{%
781 \edef\ZREF@pagenum@this{%
782 \zref@extractdefault\ZREF@refname@this{abspage}{0}%
783 }%
784 \edef\ZREF@refname@next{%
785 \zref@thepage@name{%
786 \the\numexpr\ZREF@pagenum@this+1%
787 }%
788 }%
789 \ifnum\ZREF@pagenum@this>0 %
790 \ZREF@ifrefundefined{LastPage}{%
791 \zref@ifrefundefined\ZREF@refname@next{%
792 }{%
793 \chardef\ZREF@call=2 % next page
794 }%
795 }{%
796 \edef\ZREF@pagenum@last{%
797 \zref@extractdefault{LastPage}{abspage}{0}%

```

```

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

```

## 6.9 Module `totpages`

```

831 <*totpages>
832 \NeedsTeXFormat{LaTeX2e}
833 \ProvidesPackage{zref-totpages}%
834 [2010/03/26 v2.8 Module totpages for zref (HO)]%
835 \RequirePackage{zref-base}[2010/03/26]
836 \@ifundefined{ZREF@baseok}{\endinput}{}

```

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

```

837 \RequirePackage{zref-abspage}[2010/03/26]
838 \RequirePackage{zref-lastpage}[2010/03/26]

```

`\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.

```

839 \newcommand*{\ztotpages}{%
840 \zref@extractdefault{LastPage}{abspage}{0}%
841 }

```

Also we mark the reference `LastPage` as used:

```

842 \AtBeginDocument{%
843 \zref@refused{LastPage}%
844 }
845 </totpages>

```

## 6.10 Module `runs`

This module does not use the label-reference-system. The reference changes with each  $\text{\LaTeX}$  run and would force a rerun warning always.

```

846 <*runs>
847 \NeedsTeXFormat{LaTeX2e}
848 \ProvidesPackage{zref-runs}%
849 [2010/03/26 v2.8 Module runs for zref (HO)]%

```

\zruns

```

850 \providecommand*{\zruns}{0}%
851 \AtBeginDocument{%
852   \edef\zruns{\number\numexpr\zruns+1}%
853   \begingroup
854     \def\on@line{}%
855     \PackageInfo{zref-runs}{LaTeX runs: \zruns}%
856     \if@files
857       \immediate\write\@mainaux{%
858         \string\gdef\string\zruns{\zruns}%
859       }%
860     \fi
861   \endgroup
862 }

863 </runs>

```

## 6.11 Module perpage

```

864 <*perpage>
865 \NeedsTeXFormat{LaTeX2e}
866 \ProvidesPackage{zref-perpage}%
867 [2010/03/26 v2.8 Module perpage for zref (HO)]%
868 \RequirePackage{zref-base}[2010/03/26]
869 \@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.

```
870 \RequirePackage{zref-abspage}[2010/03/26]
```

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

```

871 \zref@newprop*{pagevalue}[0]{\number\c@page}
872 \zref@newlist{perpage}
873 \zref@addprop{perpage}{abspage}
874 \zref@addprop{perpage}{page}
875 \zref@addprop{perpage}{pagevalue}

```

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

```
876 \newcounter{zpage}
```

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

```
877 \zref@requireunique
```

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`.

```

878 \let\ZREF@org@stepcounter\stepcounter
879 \def\stepcounter#1{%
880   \ifcsname @stepcounterhook@#1\endcsname
881     \csname @stepcounterhook@#1\endcsname
882   \fi
883   \ZREF@org@stepcounter{#1}%
884 }

```

\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.

```
885 \newcommand*{\zmakeperpage}{%
886   \@ifnextchar[\ZREF@makeperpage@opt{\ZREF@makeperpage[\z@]}%
887 }
```

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

```
888 \def\ZREF@makeperpage@opt[#1]{%
889   \begingroup
890     \edef\x{\endgroup
891       \noexpand\ZREF@makeperpage[\number\numexpr#1-1\relax]%
892     }%
893   \x
894 }

895 \def\ZREF@makeperpage[#1]#2{%
896   \@ifundefined{@stepcounterhook@#2}{%
897     \expandafter\gdef\csname @stepcounterhook@#2\endcsname{%
898     }%
899     \expandafter\gdef\csname ZREF@perpage@#2\endcsname{%
900       \ZREF@perpage@step{#2}{#1}%
901     }%
902     \expandafter\g@addto@macro\csname @stepcounterhook@#2\endcsname{%
903       \ifcsname ZREF@perpage@#2\endcsname
904         \csname ZREF@perpage@#2\endcsname
905       \fi
906     }%
907 }
```

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

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

First the reference is generated.

```
909   \global\advance\c@zref@unique\@ne
910   \begingroup
911     \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.

```
912     \zref@ifrefundefined\thezref@unique{%
913       \global\c@zpage=\c@page
914       \global\let\thezpage\thepage
915       \expandafter\xdef\csname ZREF@abspage@#1\endcsname{\number\c@abspage}%
916     }{%
```

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

```
917       \global\c@zpage=\zref@extract\thezref@unique{pagevalue}\relax
918       \xdef\thezpage{\noexpand\zref@extract{\thezref@unique}{page}}%
919       \expandafter\xdef\csname ZREF@abspage@#1\endcsname{%
920         \zref@extractdefault\thezref@unique{abspage}{\number\c@abspage}%
921       }%
922     }%
```

Page changes are detected by a changed absolute page number.

```
923     \expandafter\ifx\csname ZREF@abspage@#1\endcsname
924       \csname ZREF@currentabspage@#1\endcsname
925   \else
926     \global\csname c@#1\endcsname=#2\relax
927     \global\expandafter\let
928       \csname ZREF@currentabspage@#1\endcsname
929       \csname ZREF@abspage@#1\endcsname
```

```

930 \fi
931 \endgroup
932 }

```

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

```

933 \newcommand*{\zunmakeperpage}[1]{%
934 \global\expandafter\let\csname ZREF@perpage@#1\endcsname\@undefined
935 }

936 </perpage>

```

## 6.12 Module `titleref`

```

937 <*titleref>
938 \NeedsTeXFormat{LaTeX2e}
939 \ProvidesPackage{zref-titleref}%
940 [2010/03/26 v2.8 Module titleref for zref (HO)]%
941 \RequirePackage{zref-base}[2010/03/26]
942 \@ifundefined{ZREF@baseok}{\endinput}{}
943 \RequirePackage{getttitlestring}[2009/12/08]

```

### 6.12.1 Implementation

```

944 \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`.

```

945 \let\zref@titleref@current\@empty

```

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

```

946 \zref@newprop{title}{\zref@titleref@current}%
947 \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.

```

948 \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`.

```

949 \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.

```

950 \def\zref@titleref@cleanup#1{%
951 \begingroup
952 \toks@\expandafter{%
953 \ZREF@titleref@hook
954 #1%
955 }%

```

```

956 \expandafter\endgroup
957 \expandafter\def\expandafter\ZREF@titleref@hook\expandafter{%
958   \the\toks@
959 }%
960 }%

```

`\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.

```

961 \newif\ifzref@titleref@stripperperiod
962 \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.

```

963 \def\zref@titleref@setcurrent#1{%
964   \ifzref@titleref@expand
965     \GetTitleStringExpand{#1}%
966   \else
967     \GetTitleStringNonExpand{#1}%
968   \fi
969   \edef\zref@titleref@current{%
970     \detokenize\expandafter{\GetTitleStringResult}%
971   }%
972   \ifzref@titleref@stripperperiod
973     \edef\zref@titleref@current{%
974       \expandafter\ZREF@stripperperiod\zref@titleref@current
975       \@empty.\@empty\@nil
976     }%
977   \fi
978 }%
979 \GetTitleStringDisableCommands{%
980   \ZREF@titleref@hook
981 }

```

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

```

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

```

### 6.12.2 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`.

```

983 \define@key{ZREF@TR}{expand}[true]{%
984   \csname zref@titleref@expand#1\endcsname
985 }%
986 \define@key{ZREF@TR}{stripperperiod}[true]{%
987   \csname zref@titleref@stripperperiod#1\endcsname
988 }%
989 \define@key{ZREF@TR}{cleanup}{%
990   \zref@titleref@cleanup{#1}%
991 }%
992 \define@key{ZREF@TR}{title}{%
993   \def\zref@titleref@current{#1}%
994 }%
995 \newcommand*\ztitlerefsetup{%
996   \setkeys{ZREF@TR}%
997 }%

```

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

```

998 \newcommand*\ztitleref{%
999   \zref@wrapper@babel\ZREF@titleref
1000 }%
1001 \def\ZREF@titleref#1{%
1002   \begingroup
1003     \zref@refused{#1}%
1004     \let\label\ltx@gobble
1005     \zref@extract{#1}{title}%
1006   \endgroup
1007 }%

```

### 6.12.3 Patches for section and caption commands

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

Captions of figures and tables.

```

1008 \AtBeginDocument{%
1009   \ZREF@patch{@caption}{%
1010     \long\def\@caption#1[#2]{%
1011       \zref@titleref@setcurrent{#2}%
1012       \ZREF@org@@caption{#1}[{#2}]%
1013     }%
1014   }%

```

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

```

1015   \ZREF@patch{@part}{%
1016     \def\@part[#1]{%
1017       \zref@titleref@setcurrent{#1}%
1018       \ZREF@org@@part[{#1}]%
1019     }%
1020   }%
1021   \ZREF@patch{@chapter}{%
1022     \def\@chapter[#1]{%
1023       \zref@titleref@setcurrent{#1}%
1024       \ZREF@org@@chapter[{#1}]%
1025     }%
1026   }%
1027   \ZREF@patch{@sect}{%
1028     \def\@sect#1#2#3#4#5#6[#7]{%
1029       \zref@titleref@setcurrent{#7}%
1030       \ZREF@org@@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}]%
1031     }%
1032   }%

```

The star versions of the section commands.

```

1033   \ZREF@patch{@spart}{%
1034     \def\@spart#1{%
1035       \zref@titleref@setcurrent{#1}%
1036       \ZREF@org@@spart{#1}%
1037     }%
1038   }%
1039   \ZREF@patch{@schapter}{%
1040     \def\@schapter#1{%
1041       \zref@titleref@setcurrent{#1}%
1042       \ZREF@org@@schapter{#1}%
1043     }%
1044   }%
1045   \ZREF@patch{@ssect}{%
1046     \def\@ssect#1#2#3#4#5{%
1047       \zref@titleref@setcurrent{#5}%
1048       \ZREF@org@@ssect{#1}{#2}{#3}{#4}{#5}%
1049     }%
1050   }%

```

Class beamer.

```

1051 \@ifclassloaded{beamer}{%
1052   \ZREF@patch{beamer@section}{%
1053     \long\def\beamer@section[#1]{%
1054       \zref@titleref@setcurrent{#1}%
1055       \ZREF@org@beamer@section[{#1}]%
1056     }%
1057   }%
1058   \ZREF@patch{beamer@subsection}{%
1059     \long\def\beamer@subsection[#1]{%
1060       \zref@titleref@setcurrent{#1}%
1061       \ZREF@org@beamer@subsection[{#1}]%
1062     }%
1063   }%
1064   \ZREF@patch{beamer@subsubsection}{%
1065     \long\def\beamer@subsubsection[#1]{%
1066       \zref@titleref@setcurrent{#1}%
1067       \ZREF@org@beamer@subsubsection[{#1}]%
1068     }%
1069   }%
1070 }{}%
```

Package titlesec.

```

1071 \@ifpackageloaded{titlesec}{%
1072   \ZREF@patch{ttl@sect@i}{%
1073     \def\ttl@sect@i#1#2[#3]#4{%
1074       \zref@titleref@setcurrent{#4}%
1075       \ZREF@org@ttl@sect@i{#1}{#2}[{#3}]{#4}%
1076     }%
1077   }%
1078 }{}%
```

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

```

1079 \@ifpackageloaded{longtable}{%
1080   \ZREF@patch{LT@c@ption}{%
1081     \def\LT@c@ption#1[#2]#3{%
1082       \ZREF@org@LT@c@ption{#1}[{#2}]{#3}%
1083       \zref@titleref@setcurrent{#2}%
1084     }%
1085   }%
1086 }{}%
```

Package listings: support for its caption.

```

1087 \@ifpackageloaded{listings}{%
1088   \ZREF@patch{lst@MakeCaption}{%
1089     \def\lst@MakeCaption{%
1090       \ifx\lst@label\@empty
1091       \else
1092         \expandafter\zref@titleref@setcurrent\expandafter{%
1093           \lst@@caption
1094         }%
1095       \fi
1096       \ZREF@org@lst@MakeCaption
1097     }%
1098   }%
1099 }{}%
1100 }
1101 \</titleref>
```

## 6.13 Module xr

```
1102 \<*xr>
```



```

1103 \NeedsTeXFormat{LaTeX2e}
1104 \ProvidesPackage{zref-xr}%
1105 [2010/03/26 v2.8 Module xr for zref (HO)]%
1106 \RequirePackage{zref-base}[2010/03/26]
1107 \@ifundefined{ZREF@baseok}{\endinput}{%
1108 \RequirePackage{keyval}

```

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

```
1109 \zref@newprop{url}{}%

```

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

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

```

1110 \providecommand*{\zref@xr@ext}{%
1111 \ltx@ifundefined{XR@ext}{pdf}{\XR@ext}%
1112 }%

```

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

```
1113 \newif\ifZREF@xr@zreflabel

```

`\zexternaldocument` 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).

```

1114 \newcommand*{\zexternaldocument}{%
1115 \begingroup
1116 \csname @safe@actives@true\endcsname
1117 \makeatletter
1118 \@ifstar{%
1119 \ZREF@xr@zreflabelfalse
1120 \@testopt\ZREF@xr@externaldocument}{%
1121 }{%
1122 \ZREF@xr@zreflabeltrue
1123 \@testopt\ZREF@xr@externaldocument}{%
1124 }%
1125 }%

```

If the `\include` featur 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.

```

1126 \def\ZREF@xr@externaldocument[#1]#2{%
1127 \def\ZREF@xr@prefix{#1}%
1128 \let\ZREF@xr@filelist\@empty
1129 \edef\ZREF@xr@file{#2.aux}%
1130 \filename@parse{#2}%
1131 \@testopt\ZREF@xr@graburl{#2.\zref@xr@ext}%
1132 }%
1133 \def\ZREF@xr@graburl[#1]{%
1134 \edef\ZREF@xr@url{#1}%
1135 \ZREF@xr@checkfile
1136 \endgroup
1137 }%

```

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

```
1138 \def\ZREF@xr@checkfile{%

```

```

1139 \openin\@inputcheck\ZREF@xr@file\relax
1140 \ifeof\@inputcheck
1141   \PackageWarning{zref-xr}{%
1142     File '\ZREF@xr@file' not found or empty,\MessageBreak
1143     labels not imported%
1144   }%
1145 \else
1146   \PackageInfo{zref-xr}{%
1147     Label \ifZREF@xr@zreflabel (zref) \fi import from '\ZREF@xr@file'%
1148   }%
1149   \def\ZREF@xr@found{0}%
1150   \def\ZREF@xr@ignored{0}%
1151   \ZREF@xr@processfile
1152   \closein\@inputcheck
1153   \begingroup
1154     \let\on@line\@empty
1155     \PackageInfo{zref-xr}{%
1156       Statistics for '\ZREF@xr@file': %
1157       \ZREF@xr@found\space found, %
1158       \ZREF@xr@ignored\space ignored%
1159     }%
1160   \endgroup
1161 \fi
1162 \ifx\ZREF@xr@filelist\@empty
1163 \else
1164   \edef\ZREF@xr@file{\expandafter\@car\ZREF@xr@filelist\@nil}%
1165   \edef\ZREF@xr@filelist{\expandafter\@cdr\ZREF@xr@filelist\@nil}%
1166   \expandafter\ZREF@xr@checkfile
1167 \fi
1168 }%

```

\ZREF@xr@processfile

```

1169 \def\ZREF@xr@processfile{%
1170   \read\@inputcheck to\ZREF@xr@line
1171   \expandafter\ZREF@xr@processline\ZREF@xr@line..\ZREF@nil
1172   \ifeof\@inputcheck
1173   \else
1174     \expandafter\ZREF@xr@procesfile
1175   \fi
1176 }%

```

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

```

1177 \long\def\ZREF@xr@processline#1#2#3\ZREF@nil{%
1178   \def\x{#1}%
1179   \toks@{#2}%
1180   \ifZREF@xr@zreflabel
1181     \ifx\x\ZREF@xr@zref@newlabel
1182       \expandafter\ZREF@xr@process@zreflabel\ZREF@xr@line...\ZREF@nil
1183     \fi
1184   \else
1185     \ifx\x\ZREF@xr@newlabel
1186       \expandafter\ZREF@xr@process@label\ZREF@xr@line...[]\ZREF@nil
1187     \fi
1188   \fi
1189   \ifx\x\ZREF@xr@@input
1190     \edef\ZREF@xr@filelist{%
1191       \etex@unexpanded\expandafter{\ZREF@xr@filelist}%
1192       {\filename@area\the\toks@}%
1193     }%
1194   \fi
1195   \ifeof\@inputcheck
1196   \else

```

```

1197 \expandafter\ZREF@xr@processfile
1198 \fi
1199 }%
1200 \def\ZREF@xr@process@zreflabel\zref@newlabel#1#2#3\ZREF@nil{%
1201 \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1202 \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1203 \def\x{#2}%
1204 \@ifundefined{\ZREF@xr@refname}{%
1205 \let\ZREF@xr@list\x
1206 \ifx\ZREF@xr@list\@empty
1207 \PackageWarningNoLine{zref-xr}{%
1208 Label ‘#1’ without properties ignored\MessageBreak
1209 in file ‘\ZREF@xr@file’%
1210 }%
1211 \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
1212 \else
1213 \expandafter\ZREF@xr@checklist\x\ZREF@nil
1214 \expandafter\global\expandafter\let
1215 \csname \ZREF@xr@refname\endcsname\x
1216 \fi
1217 \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1218 }{%
1219 \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
1220 }%
1221 }%
1222 \def\ZREF@xr@process@label\newlabel#1#2#3[#4]#5\ZREF@nil{%
1223 \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1224 \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1225 \def\x{#2}%
1226 \@ifundefined{\ZREF@xr@refname}{%
1227 \expandafter\ZREF@xr@scanparams
1228 \csname\ZREF@xr@refname\expandafter\endcsname
1229 \x{}{}{}{}{}\ZREF@nil
1230 \ifx\\#4\\%
1231 \else
1232 % ntheorem knows an optional argument at the end of \newlabel
1233 \zref@ifpropundefined{theotype}{%
1234 \zref@newprop{theotype}{}%
1235 }{}%
1236 \expandafter\g@addto@macro
1237 \csname\ZREF@xr@refname\endcsname{\theotype{#4}}%
1238 \fi
1239 \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1240 }{%
1241 \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
1242 }%
1243 }
1244 \def\ZREF@xr@zref@newlabel{\zref@newlabel}%
1245 \def\ZREF@xr@newlabel{\newlabel}%
1246 \def\ZREF@xr@input{\@input}%

```

\ZREF@xr@ignorewarning

```

1247 \def\ZREF@xr@ignorewarning#1{%
1248 \PackageWarningNoLine{zref-xr}{%
1249 Label ‘#1’ is already in use\MessageBreak
1250 in file ‘\ZREF@xr@file’%
1251 }%
1252 \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
1253 }%

```

\ZREF@xr@checklist

```

1254 \def\ZREF@xr@checklist#1#2#3\ZREF@nil{%

```

```

1255 \ifx\@undefined#1\relax
1256 \expandafter\ZREF@xr@checkkey\string#1\@nil
1257 \fi
1258 \ifx\#3\%
1259 \else
1260 \@ReturnAfterFi{%
1261 \ZREF@xr@checklist#3\ZREF@nil
1262 }%
1263 \fi
1264 }%
1265 \long\def\@ReturnAfterFi#1\fi{\fi#1}%
1266 \def\ZREF@xr@checkkey#1#2\@nil{%
1267 \zref@ifpropundefined{#2}{%
1268 \zref@newprop{#2}{}%
1269 }{}%
1270 }%

```

\ZREF@xr@scanparams

```

1271 \def\ZREF@xr@scanparams#1#2#3#4#5#6#7\ZREF@nil{%
1272 \global\let#1\@empty
1273 \ZREF@foundfalse
1274 \ZREF@xr@scantitleref#1#2\TR@TitleReference{}{}\ZREF@nil
1275 \ifZREF@found
1276 \else
1277 \g@addto@macro#1{\default{#2}}%
1278 \fi
1279 % page
1280 \g@addto@macro#1{\page{#3}}%
1281 % nameref title
1282 \ifZREF@found
1283 \else
1284 \ifx\#4\%
1285 \else
1286 \zref@ifpropundefined{title}{%
1287 \zref@newprop{title}{}%
1288 }{}%
1289 \g@addto@macro#1{\title{#4}}%
1290 \fi
1291 \fi
1292 % anchor
1293 \ifx\#5\%
1294 \else
1295 \zref@ifpropundefined{anchor}{%
1296 \zref@newprop{anchor}{}%
1297 }{}%
1298 \g@addto@macro#1{\anchor{#5}}%
1299 \fi
1300 \ifx\#6\%
1301 \else
1302 \zref@ifpropundefined{url}{%
1303 \zref@newprop{url}{}%
1304 }{}%
1305 \g@addto@macro#1{\url{#6}}%
1306 \fi
1307 }%

```

\ZREF@xr@scantitleref

```

1308 \def\ZREF@xr@scantitleref#1#2\TR@TitleReference#3#4#5\ZREF@nil{%
1309 \ifx\#5\%
1310 \else
1311 \g@addto@macro#1{%
1312 \default{#3}%

```

```

1313     \title{#4}%
1314 }%
1315 \ZREF@foundtrue
1316 \fi
1317 }%

```

`\ZREF@xr@urlcheck`

```

1318 \def\ZREF@xr@urlcheck#1{%
1319   \zref@ifrefcontainsprop{#1}{anchor}{%
1320     \zref@ifrefcontainsprop{#1}{url}{%
1321       }{%
1322         \expandafter\g@addto@macro\csname Z@R@#1\expandafter\endcsname
1323         \expandafter{%
1324           \expandafter\url\expandafter{\ZREF@xr@url}%
1325         }%
1326       }%
1327     }{%
1328     }%
1329 }%

```

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

```

1330 \define@key{ZREF@XR}{ext}{%
1331   \def\zref@xr@ext{#1}%
1332 }%
1333 \newcommand*{\zxrsetup}{%
1334   \setkeys{ZREF@XR}%
1335 }%
1336 \</xr>

```

## 6.14 Module `hyperref`

UNFINISHED :-(

```

1337 \<*hyperref>
1338 \NeedsTeXFormat{LaTeX2e}
1339 \ProvidesPackage{zref-hyperref}%
1340 [2010/03/26 v2.8 Module hyperref for zref (H0)]%
1341 \RequirePackage{zref-base}[2010/03/26]
1342 \@ifundefined{ZREF@baseok}{\endinput}{}
1343 \zref@newprop{anchor}[]{%
1344   \ltx@ifundefined{@currentHref}{\@currentHref}%
1345 }%
1346 \zref@addprop\ZREF@mainlist{anchor}%
1347 \</hyperref>

```

## 6.15 Module `savepos`

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

### 6.15.1 Identification

```

1348 \<*savepos>
1349 \NeedsTeXFormat{LaTeX2e}
1350 \ProvidesPackage{zref-savepos}%
1351 [2010/03/26 v2.8 Module savepos for zref (H0)]%
1352 \RequirePackage{zref-base}[2010/03/26]
1353 \@ifundefined{ZREF@baseok}{\endinput}{}

```

### 6.15.2 Availability

First we check, whether the feature is available.

```

1354 \ltx@ifundefined{pdfsavepos}{%
1355   \ZREF@ErrorNoLine{%
1356     \string\pdfsavepos\space is not supported.\MessageBreak
1357     It is provided by pdfTeX (1.40) or XeTeX%
1358   }\ZREF@UpdatePdfTeX
1359   \endinput
1360 }{}%

```

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`.

```

1361 \RequirePackage{ifpdf}
1362 \ifpdf
1363 \else
1364   \begingroup\expandafter\expandafter\expandafter\endgroup
1365   \expandafter\ifx\csname pdftexversion\endcsname\relax
1366   \else
1367     \ifnum\pdftexversion<140 %
1368       \ZREF@ErrorNoLine{%
1369         \string\pdfsavepos\space is not supported in DVI mode\MessageBreak
1370         of this pdfTeX version%
1371       }\ZREF@UpdatePdfTeX
1372       \expandafter\expandafter\expandafter\endinput
1373     \fi
1374   \fi
1375 \fi

```

### 6.15.3 Setup

```

1376 \zref@newlist{savepos}
1377 \zref@newprop*{posx}[0]{\the\pdflastxpos}
1378 \zref@newprop*{posy}[0]{\the\pdflastypos}
1379 \zref@addprop{savepos}{posx}
1380 \zref@addprop{savepos}{posy}

```

### 6.15.4 User macros

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

```

1381 \def\zsavapos#1{%
1382   \@bsphack
1383   \if@filesw
1384     \pdfsavepos
1385     \zref@labelbylist{#1}{savepos}%
1386   \fi
1387   \@esphack
1388 }

```

`\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.

```

1389 \newcommand*{\zposx}[1]{%
1390   \zref@extract{#1}{posx}%
1391 }%
1392 \newcommand*{\zposy}[1]{%
1393   \zref@extract{#1}{posy}%
1394 }%

```

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.

```

1395 </savepos>

```

## 6.16 Module dotfill

```

1396 <*dotfill>
1397 \NeedsTeXFormat{LaTeX2e}
1398 \ProvidesPackage{zref-dotfill}%
1399 [2010/03/26 v2.8 Module dotfill for zref (HO)]%
1400 \RequirePackage{zref-base}[2010/03/26]
1401 \@ifundefined{ZREF@baseok}{\endinput}{}

```

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

```
1402 \RequirePackage{zref-savepos}[2010/03/26]
```

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

```
1403 \zref@require@unique
```

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

```
1404 \RequirePackage{keyval}
```

The definitions of the keys follow.

```

1405 \define@key{ZREF@DF}{unit}{%
1406   \def\ZREF@df@unit{#1}%
1407 }
1408 \define@key{ZREF@DF}{min}{%
1409   \def\ZREF@df@min{#1}%
1410 }
1411 \define@key{ZREF@DF}{dot}{%
1412   \def\ZREF@df@dot{#1}%
1413 }

```

Defaults are set, see user interface.

```

1414 \providecommand\ZREF@df@min{2}
1415 \providecommand\ZREF@df@unit{.44em}
1416 \providecommand\ZREF@df@dot{.}

```

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

```
1417 \newcommand*{\zdotfillsetup}{\setkeys{ZREF@DF}}
```

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

```

1418 \newcommand*{\zdotfill}{%
1419   \leavevmode
1420   \global\advance\c@zref@unique\@ne
1421   \begingroup
1422     \def\ZREF@temp{zref@\number\c@zref@unique}%
1423     \pdfsavepos
1424     \zref@labelbyprops{\thezref@unique L}{posx}%
1425     \setlength{\dimen@}{\ZREF@df@unit}%
1426     \zref@ifrefundefined{\thezref@unique R}{%
1427       \ZREF@dotfill
1428     }{%
1429       \ifnum\numexpr\zposx{\thezref@unique R}-\zposx{\thezref@unique L}\relax
1430         <\dimexpr\ZREF@df@min\dimen@\relax
1431         \hfill
1432       \else
1433         \ZREF@dotfill
1434       \fi
1435     }%
1436     \pdfsavepos
1437     \zref@labelbyprops{\thezref@unique R}{posx}%
1438   \endgroup
1439   \kern\z@
1440 }

```

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

```

1441 \def\ZREF@dotfill{%
1442   \cleaders\hb@xt@\dimen@{\hss\ZREF@df@dot\hss}\hfill
1443 }
1444 </dotfill>

```

## 7 Test

### 7.1 \zref@localaddprop

```

1445 <*test1>
1446 \NeedsTeXFormat{LaTeX2e}
1447 \nofiles
1448 \documentclass{article}
1449 \usepackage{zref-base}[2010/03/26]
1450 \usepackage{qstest}
1451 \IncludeTests{*}
1452 \LogTests{log}{*}{*}
1453
1454 \makeatletter
1455 \begin{qstest}{localaddprop}{localaddprop}
1456   \Expect*{\Z@L@main}*{{default}{page}}%
1457   \zref@newprop{foobar}{F00}%
1458   \zref@newlist{alist}%
1459   \Expect*{\Z@L@alist}{}%
1460   \begingroup
1461     \zref@localaddprop{main}{foobar}%
1462     \Expect*{\Z@L@main}{{default}{page}{foobar}}%
1463     \zref@localaddprop{alist}{page}%
1464     \Expect*{\Z@L@alist}{{page}}%
1465   \endgroup
1466   \Expect*{\Z@L@main}*{{default}{page}}%
1467   \Expect*{\Z@L@alist}{}%
1468 \end{qstest}
1469 @@end
1470 </test1>

```

### 7.2 Module runs

```

1471 <*test-runs>
1472 \NeedsTeXFormat{LaTeX2e}
1473 \documentclass{article}
1474 \usepackage{zref-runs}[2010/03/26]
1475 \usepackage{qstest}
1476 \IncludeTests{*}
1477 \LogTests{log}{*}{*}
1478
1479 \begin{qstest}{zruns-preamble}{zruns-preamble}
1480   \Expect{0}*{\zruns}%
1481 \end{qstest}
1482
1483 \AtBeginDocument{%
1484   \begin{qstest}{zruns-atbegindocument}{zruns-atbegindocument}%
1485     \Expect*{\number\ExpectRuns}*{\zruns}%
1486   \end{qstest}%
1487 }
1488
1489 \begin{document}
1490 \begin{qstest}{zruns-document}{zruns-document}
1491   \Expect*{\number\ExpectRuns}*{\zruns}%
1492 \end{qstest}
1493 \end{document}
1494 </test-runs>

```



## 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  $\text{\TeX}$  Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

### 8.2 Bundle installation

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

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

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

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

### 8.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain- $\text{\TeX}$ :

```
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):

---

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

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

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  $\text{\TeX}$  distribution (te $\text{\TeX}$ , mik $\text{\TeX}$ , ...) relies on file name databases, you must refresh these. For example, te $\text{\TeX}$  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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), 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 pdf $\text{\LaTeX}$ :

```
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  $\epsilon$ -TeX's `\unexpanded`.

[2007/05/28 v2.1]

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

[2008/09/21 v2.2]

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

[2008/10/01 v2.3]

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

[2009/08/07 v2.4]

- Module `runs` added.

[2009/12/06 v2.5]

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

[2009/12/07 v2.6]

- Version date added for package atveryend.

[2009/12/08 v2.7]

- Module titleref: Use of package gettitlestring.

[2010/03/26 v2.8]

- \zifrefundefined added.
- Module lastpage: Macros \zref@iflastpage and \ziflastpage added.
- Module thepage added.
- Module nextpage 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	
\@@end .....	1469
\@ReturnAfterFi .....	1260, 1265
\@addtoreset .....	585, 645
\@auxout .....	453
\@bsphack .....	409, 419, 1382
\@caption .....	1010
\@car .....	1164
\@cdr .....	1165
\@chapter .....	1022
\@currentHref .....	1344
\@currentlabel .....	604
\@ehc .....	275, 289, 373
\@empty .....	221, 269, 381, 457, 531, 608, 945, 949, 975, 982, 1090, 1128, 1154, 1162, 1206, 1272
\@esphack .....	416, 436, 1387
\@for .....	423
\@ifclassloaded .....	1051
\@ifdefinable .....	268
\@ifnextchar .....	388, 886
\@ifpackageloaded ..	1071, 1079, 1087
\@ifstar .....	377, 1118
\@ifundefined .....	192, 583, 615, 642, 660, 679, 717, 748, 836, 869, 896, 942, 1107, 1204, 1226, 1342, 1353, 1401
\@input .....	1246
\@inputcheck .....	1139, 1140, 1152, 1170, 1172, 1195
\@latex@warning .....	503
\@mainaux .....	857
\@namedef .....	391
\@ne .....	909, 1420
\@newl@bel .....	264
\@nil ..	975, 982, 1164, 1165, 1256, 1266
\@part .....	1016
\@schapter .....	1040
\@sect .....	1028
\@spart .....	1034
\@ssect .....	1046
\@testopt .....	1120, 1123, 1131
\@tfor .....	299, 314, 458
\@undefined .....	934, 1255
\@ .....	27, 28, 29, 30, 153, 155, 157, 158, 170, 173, 1230, 1258, 1284, 1293, 1300, 1309
\_ .....	46, 47
A	
\AddLineBeginAux .....	260
\advance .....	684, 909, 1420
\afterassignment .....	756, 760
\AfterLastShipout .....	681
\Alph .....	7
\anchor .....	1298
\AtBeginDocument .....	663, 842, 851, 1008, 1483
\AtBeginShipout .....	649, 722
\AtEndOfPackage .....	195
B	
\beamer@section .....	1053
\beamer@subsection .....	1059
\beamer@subsubsection .....	1065
\begin .....	25, 59, 102, 108, 156, 172, 1455, 1479, 1484, 1489, 1490
\bfseries .....	597
C	
\c@abspage .....	652, 915, 920
\c@page .....	684, 871, 913
\c@zpage .....	913, 917
\c@zref@unique ..	589, 909, 1420, 1422

<code>\chapter</code> .....	26, 32, 34, 63, 84	<b>G</b>	
<code>\ChapterPages</code> .....	93, 113	<code>\g@addto@macro</code> .....	
<code>\ChapterStart</code> .....	80, 135, 150, 166	...	331, 352, 902, 1236, 1277,
<code>\ChapterStop</code> .....	87, 148, 165, 184		1280, 1289, 1298, 1305, 1311, 1322
<code>\chardef</code> .....	778, 793, 802, 806	<code>\G@refundefinedtrue</code> .....	502
<code>\cleaders</code> .....	1442	<code>\gdef</code> .....	397, 858, 897, 899
<code>\cleardoublepage</code> .....	81, 88	<code>\GetTitleStringDisableCommands</code> ..	979
<code>\clearpage</code> .....	64	<code>\GetTitleStringExpand</code> .....	965
<code>\closein</code> .....	1152	<code>\GetTitleStringNonExpand</code> .....	967
<code>\csname</code> .....	229, 232,	<code>\GetTitleStringResult</code> .....	970
	233, 269, 279, 293, 318, 331,		
	344, 363, 392, 393, 397, 400,	<b>H</b>	
	403, 413, 460, 462, 467, 468,	<code>\hb@xt@</code> .....	1442
	484, 511, 522, 523, 525, 544,	<code>\hfill</code> .....	1431, 1442
	545, 546, 570, 881, 897, 899,	<code>\hss</code> .....	1442
	902, 904, 915, 919, 923, 924,		
	926, 928, 929, 934, 984, 987,	<b>I</b>	
	1116, 1215, 1228, 1237, 1322, 1365	<code>\if@files</code> .....	448, 682, 856, 1383
<code>\current@chapid</code> .....	82, 90	<code>\if@safe@actives</code> .....	563
<b>D</b>		<code>\ifcase</code> .....	116, 814
<code>\DeclareOption</code> .....	194	<code>\ifcsname</code> .....	557, 880, 903
<code>\default</code> .....	1277, 1312	<code>\ifeof</code> .....	1140, 1172, 1195
<code>\define@key</code> .....	983, 986,	<code>\ifetex@unexpanded</code> .....	247, 349
	989, 992, 1330, 1405, 1408, 1411	<code>\ifnum</code> ..	696, 789, 799, 805, 1367, 1429
<code>\detokenize</code> .....	970	<code>\ifodd</code> .....	125
<code>\dftest</code> .....	167, 174, 175,	<code>\ifpdf</code> .....	1362
	176, 177, 178, 179, 180, 181, 182	<code>\ifx</code> ...	229, 279, 302, 363, 459, 484,
<code>\dimen@</code> .....	1425, 1430, 1442		530, 617, 923, 1090, 1162, 1181,
<code>\dimexpr</code> .....	153, 155, 1430		1185, 1189, 1206, 1230, 1255,
<code>\do</code> .....	300, 319, 423, 458		1258, 1284, 1293, 1300, 1309, 1365
<code>\documentclass</code> 4, 41, 70, 252, 1448, 1473		<code>\ifZREF@found</code> ...	226, 307, 1275, 1282
<code>\dotfill</code> .....	169, 173	<code>\ifZREF@immediate</code> ..	438, 450, 454, 460
		<code>\ifzref@titleref@expand</code> ...	948, 964
		<code>\ifzref@titleref@stripperperiod</code>	961, 972
		<code>\ifZREF@xref@zreflabel</code>	1113, 1147, 1180
<b>E</b>		<code>\immediate</code> .....	443, 857
<code>\emph</code> .....	150	<code>\IncludeTests</code> .....	1451, 1476
<code>\end</code> .....	36, 66, 130, 159, 183,	<code>\item</code> .....	109, 112, 114, 122, 126, 128
	185, 1468, 1481, 1486, 1492, 1493		
<code>\endcsname</code> .....	229, 232, 233, 269, 279,	<b>K</b>	
	293, 318, 331, 344, 363, 392,	<code>\kern</code> .....	1439
	393, 397, 400, 403, 413, 460,		
	462, 467, 468, 484, 511, 522,	<b>L</b>	
	523, 525, 544, 545, 546, 557,	<code>\l</code> .....	421
	570, 880, 881, 897, 899, 902,	<code>\label</code> .....	617, 1004
	903, 904, 915, 919, 923, 924,	<code>\leavevmode</code> .....	1419
	926, 928, 929, 934, 984, 987,	<code>\LogTests</code> .....	1452, 1477
	1116, 1215, 1228, 1237, 1322, 1365	<code>\lst@caption</code> .....	1093
<code>\endinput</code> 192, 244, 257, 615, 642, 660,		<code>\lst@label</code> .....	1090
	679, 717, 748, 836, 869, 942,	<code>\lst@MakeCaption</code> .....	1089
	1107, 1342, 1353, 1359, 1372, 1401	<code>\LT@caption</code> .....	1081
<code>\etex@unexpanded</code> .....	358, 551, 1191	<code>\ltx@firstofone</code> ...	234, 553, 555, 558
<code>\Expect</code> .....	1456, 1459, 1462,	<code>\ltx@firstoftwo</code> .....	
	1464, 1466, 1467, 1480, 1485, 1491		280, 308, 364, 485, 534, 566, 698
<code>\ExpectRuns</code> .....	1485, 1491	<code>\ltx@gobble</code> .....	230, 617, 618, 1004
		<code>\ltx@gobbletwo</code> .....	585, 645
<b>F</b>		<code>\ltx@ifundefined</code> .....	237, 727, 1354
<code>\fancyhead</code> .....	53, 56	<code>\ltx@ifundefined</code> .....	1111, 1344
<code>\fancyhf</code> .....	52, 55	<code>\ltx@secondoftwo</code> .....	282, 310,
<code>\fancypagestyle</code> .....	54		366, 487, 519, 532, 560, 564, 700
<code>\filename@area</code> .....	1192	<code>\ltx@space</code> .....	799, 805
<code>\filename@parse</code> .....	1130		
<code>\foo</code> .....	20, 31, 33, 35	<b>M</b>	
<code>\frontmatter</code> .....	60, 105	<code>\m@ne</code> .....	684

<code>\mainmatter</code> . . . . .	62, 134	<code>\reset@font</code> . . . . .	597
<code>\makeatletter</code> . . . . .	11, 76, 103, 1117, 1454	<code>\rightarrow</code> . . . . .	47
<code>\makeatother</code> . . . . .	18, 101		
<code>\makebox</code> . . . . .	169, 170		
<code>\MessageBreak</code> . . . . .	250, 1142, 1208, 1249, 1356, 1369		
	<b>N</b>		<b>S</b>
<code>\NeedsTeXFormat</code> . . . . .	3, 188, 214, 611, 638, 656, 673, 713, 744, 832, 847, 865, 938, 1103, 1338, 1349, 1397, 1446, 1472	<code>\section</code> . . . . .	65, 137, 145
<code>\newcommand</code> . . . . .	20, 80, 87, 93, 167, 490, 616, 623, 632, 635, 703, 738, 755, 769, 772, 773, 839, 885, 933, 995, 998, 1114, 1333, 1389, 1392, 1417, 1418	<code>\setcounter</code> . . . . .	648
<code>\newcounter</code> . . . . .	6, 586, 646, 876	<code>\setkeys</code> . . . . .	996, 1334, 1417
<code>\newif</code> . . . . .	226, 438, 948, 961, 1113	<code>\setlength</code> . . . . .	1425
<code>\newlabel</code> . . . . .	1222, 1232, 1245	<code>\space</code> . . . . .	504, 1157, 1158, 1356, 1369
<code>\newpage</code> . . . . .	143	<code>\stepcounter</code> . . . . .	21, 650, 878, 879
<code>\nfss@text</code> . . . . .	597		
<code>\nofiles</code> . . . . .	1447		
<code>\number</code> . . . . .	96, 111, 589, 728, 852, 871, 891, 915, 920, 1422, 1485, 1491		
<code>\numexpr</code> . . . . .	96, 111, 116, 730, 786, 852, 891, 1202, 1211, 1224, 1252, 1429		
	<b>O</b>		<b>T</b>
<code>\on@line</code> . . . . .	221, 854, 1154	<code>\tableofcontents</code> . . . . .	61, 132
<code>\openin</code> . . . . .	1139	<code>\the</code> . . . . .	13, 153, 155, 430, 435, 466, 480, 652, 690, 724, 730, 786, 958, 1192, 1202, 1211, 1224, 1252, 1377, 1378
		<code>\thechapter</code> . . . . .	14
		<code>\thefoo</code> . . . . .	7, 12, 22
		<code>\thetype</code> . . . . .	1237
		<code>\thepage</code> . . . . .	45, 46, 47, 451, 455, 504, 605, 914
		<code>\thezpage</code> . . . . .	14, 914, 918
		<code>\thezref@unique</code> . . . . .	9, 588, 911, 912, 917, 918, 920, 1424, 1426, 1429, 1437
		<code>\title</code> . . . . .	1289, 1313
		<code>\toks@</code> . . . . .	422, 429, 430, 435, 464, 466, 479, 480, 685, 690, 952, 958, 1179, 1192
		<code>\TR@TitleReference</code> . . . . .	1274, 1308
		<code>\ttl@sect@i</code> . . . . .	1073
			<b>U</b>
		<code>\unexpanded</code> . . . . .	250, 255
		<code>\UniqueCounterCall</code> . . . . .	770
		<code>\UniqueCounterNew</code> . . . . .	753
		<code>\url</code> . . . . .	1305, 1324
		<code>\usepackage</code> . . . . .	9, 43, 50, 72, 74, 1449, 1450, 1474, 1475
			<b>V</b>
		<code>\value</code> . . . . .	13, 724
		<code>\verb</code> . . . . .	173
			<b>W</b>
		<code>\write</code> . . . . .	442, 443, 857
			<b>X</b>
		<code>\x</code> . . . . .	299, 301, 302, 423, 424, 426, 430, 571, 573, 812, 828, 890, 893, 1178, 1181, 1185, 1189, 1203, 1205, 1213, 1215, 1225, 1229
		<code>\XR@ext</code> . . . . .	1111
			<b>Y</b>
		<code>\y</code> . . . . .	298, 302
			<b>Z</b>
		<code>\z@</code> . . . . .	886, 1439
		<code>\Z@D@page</code> . . . . .	773
		<code>\Z@L@alist</code> . . . . .	1459, 1464, 1467
		<code>\Z@L@LastPage</code> . . . . .	687
		<code>\Z@L@main</code> . . . . .	686, 1456, 1462, 1466
		<code>\zdotfill</code> . . . . .	15, 170, 173, 1418

<code>\zdotfillsetup</code>	15, 1417	<code>\zref@labelbylist</code>	6, 406, 408, 724, 911, 1385
<code>\zexternaldocument</code>	16, 1114	<code>\zref@labelbyprops</code>	6, 90, 418, 777, 1424, 1437
<code>\ziflastpage</code>	10, 703	<code>\zref@listexists</code>	5, 285, 324, 337, 410
<code>\zifrefundefined</code>	7, 490	<code>\zref@listforloop</code>	313
<code>\zlabel</code>	9, 85, 106, 138, 146, 616	<code>\zref@localaddprop</code>	5, 336, 1461, 1463
<code>\zmakeperpage</code>	13, 885	<code>\ZREF@mainlist</code>	406, 600, 603, 606, 607, 653, 662, 947, 1346
<code>\znextpage</code>	12, 53, 56, 769	<code>\ZREF@makeperpage@opt</code>	886, 888
<code>\znextpagesetup</code>	12, 44, 755	<code>\ZREF@name</code>	218, 222, 273, 287, 327, 340, 371, 425
<code>\znonextpagename</code>	48, 772, 820	<code>\zref@newlabel</code>	6, 261, 263, 473, 1200, 1244
<code>\zpageref</code>	9, 127, 632	<code>\zref@newlist</code>	5, 266, 603, 680, 720, 872, 1376, 1458
<code>\zposx</code>	15, 153, 1389, 1429	<code>\ZREF@newprop</code>	379, 382, 385
<code>\zposy</code>	15, 155, 1389	<code>\zref@newprop</code>	5, 12, 13, 14, 77, 376, 604, 605, 652, 661, 871, 946, 1109, 1234, 1268, 1287, 1296, 1303, 1343, 1377, 1378, 1457
<code>\zref</code>	9, 27, 28, 29, 30, 113, 115, 124, 129, 139, 623, 633	<code>\ZREF@nextpage</code>	770, 774
<code>\ZREF@@@newprop</code>	393, 396	<code>\ZREF@nil</code>	397, 531, 546, 1171, 1177, 1182, 1186, 1200, 1213, 1222, 1229, 1254, 1261, 1271, 1274, 1308
<code>\ZREF@makeperpage</code>	886, 891, 895	<code>\ZREF@NOVALUE</code>	537
<code>\ZREF@newprop</code>	388, 390	<code>\ZREF@novalue</code>	530, 531, 537
<code>\ZREF@perpage@step</code>	900, 908	<code>\ZREF@np@call@next</code>	764, 768, 823
<code>\zref@addprop</code>	5, 15, 16, 17, 78, 323, 606, 607, 653, 662, 721, 873, 874, 875, 947, 1346, 1379, 1380	<code>\ZREF@np@call@nonext</code>	761, 767, 819
<code>\ZREF@addtoks</code>	478	<code>\ZREF@np@call@unknown</code>	757, 766, 815
<code>\ZREF@baseok</code>	608	<code>\ZREF@np@setup@i</code>	756, 759
<code>\ZREF@call</code>	778, 793, 802, 806, 814	<code>\ZREF@np@setup@ii</code>	760, 763
<code>\zref@default</code>	7, 388, 594, 596	<code>\ZREF@org@@caption</code>	1012
<code>\ZREF@df@dot</code>	1412, 1416, 1442	<code>\ZREF@org@@chapter</code>	1024
<code>\ZREF@df@min</code>	1409, 1414, 1430	<code>\ZREF@org@@part</code>	1018
<code>\ZREF@df@unit</code>	1406, 1415, 1425	<code>\ZREF@org@@schapter</code>	1042
<code>\ZREF@dotfill</code>	1427, 1433, 1441	<code>\ZREF@org@@ssect</code>	1030
<code>\ZREF@ErrorNoLine</code>	219, 238, 249, 1355, 1368	<code>\ZREF@org@@spart</code>	1036
<code>\ZREF@extract</code>	509, 514	<code>\ZREF@org@@ssect</code>	1048
<code>\zref@extract</code>	6, 97, 98, 111, 140, 508, 630, 733, 824, 917, 918, 1005, 1390, 1393	<code>\ZREF@org@beamer@section</code>	1055
<code>\zref@extractdefault</code>	6, 117, 118, 515, 538, 696, 697, 782, 797, 840, 920	<code>\ZREF@org@beamer@subsection</code>	1061
<code>\ZREF@foundfalse</code>	297, 1273	<code>\ZREF@org@beamer@subsubsection</code>	1067
<code>\ZREF@foundtrue</code>	303, 1315	<code>\ZREF@org@lst@MakeCaption</code>	1096
<code>\zref@getcurrent</code>	6, 402	<code>\ZREF@org@LT@c@ption</code>	1082
<code>\ZREF@gtmp</code>	351, 352, 353	<code>\ZREF@org@refstepcounter</code>	667
<code>\ZREF@iflastpage</code>	704, 706, 706	<code>\ZREF@org@stepcounter</code>	878, 883
<code>\zref@iflastpage</code>	10, 695, 709	<code>\ZREF@org@thepage</code>	451, 455
<code>\ZREF@iflistcontainsprop</code>	293, 295	<code>\ZREF@org@ttl@ssect@i</code>	1075
<code>\zref@iflistcontainsprop</code>	5, 292, 326, 339	<code>\ZREF@org@write</code>	442, 443
<code>\zref@iflistundefined</code>	5, 267, 278, 286	<code>\ZREF@P</code>	387, 391, 392, 393, 394, 397, 458, 460, 462, 467, 468
<code>\zref@ifpropundefined</code>	6, 362, 370, 424, 1233, 1267, 1286, 1295, 1302	<code>\ZREF@pagenum@last</code>	796, 799
<code>\ZREF@ifrefcontainsprop</code>	521, 529	<code>\ZREF@pagenum@this</code>	781, 786, 789, 799, 805
<code>\zref@ifrefcontainsprop</code>	7, 517, 1319, 1320	<code>\ZREF@patch</code>	227, 664, 1009, 1015, 1021, 1027, 1033, 1039, 1045, 1052, 1058, 1064, 1072, 1080, 1088
<code>\ZREF@ifrefundefined</code>	491, 493, 779, 790, 800	<code>\zref@prop</code>	315, 320
<code>\zref@ifrefundefined</code>	7, 483, 495, 501, 518, 539, 791, 912, 1426	<code>\zref@propexists</code>	6, 325, 338, 369, 624
<code>\ZREF@immediatetrue</code>	441	<code>\ZREF@refname@next</code>	784, 791, 800, 824
<code>\ZREF@l@addto@macro</code>	344, 349	<code>\ZREF@refname@this</code>	776, 777, 779, 782
<code>\ZREF@label</code>	411, 435, 447, 690	<code>\ZREF@refused</code>	498, 500
<code>\zref@label</code>	6, 405, 620		



<code>\zref@refused</code> . . . . .	7, 494, <a href="#">497</a> , 629, 635, 707, 708, 736, 843, 1003	<code>\ZREF@xr@file</code> . . . . .	1129, 1139, 1142, 1147, 1156, 1164, 1209, 1250
<code>\zref@require@unique</code> . . . . .	8, <a href="#">582</a> , 877, 1403	<code>\ZREF@xr@filelist</code> . . . . .	1128, 1162, 1164, 1165, 1190, 1191
<code>\zref@setcurrent</code> . . . . .	5, 83, 394, <a href="#">399</a> , 666	<code>\ZREF@xr@found</code> . . . . .	1149, 1157, 1202, 1224
<code>\zref@setdefault</code> . . . . .	7, <a href="#">593</a> , 596	<code>\ZREF@xr@graburl</code> . . . . .	1131, 1133
<code>\zref@setmainlist</code> . . . . .	7, <a href="#">599</a>	<code>\ZREF@xr@ignored</code> . . . . .	1150, 1158, 1211, 1252
<code>\ZREF@stripperperiod</code> . . . . .	974, <a href="#">982</a>	<code>\ZREF@xr@ignorewarning</code> . . . . .	1219, 1241, <a href="#">1247</a>
<code>\ZREF@temp</code> . . . . .	193, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 457, 464, 465, 473, 1422	<code>\ZREF@xr@line</code> . . . . .	1170, 1171, 1182, 1186
<code>\zref@thepage</code> . . . . .	11, <a href="#">732</a> , 740	<code>\ZREF@xr@list</code> . . . . .	1205, 1206
<code>\zref@thepage@name</code> . . . . .	11, <a href="#">727</a> , 733, 736, 785	<code>\ZREF@xr@newlabel</code> . . . . .	1185, 1245
<code>\zref@thepage@refused</code> . . . . .	735, 739	<code>\ZREF@xr@prefix</code> . . . . .	1127, 1201, 1217, 1219, 1223, 1239, 1241
<code>\ZREF@titleref</code> . . . . .	999, 1001	<code>\ZREF@xr@procesfile</code> . . . . .	1174
<code>\zref@titleref@cleanup</code> . . . . .	950, 990	<code>\ZREF@xr@process@label</code> . . . . .	1186, 1222
<code>\zref@titleref@current</code> . . . . .	945, 946, 969, 973, 974, 993	<code>\ZREF@xr@process@zreflabel</code> . . . . .	1182, 1200
<code>\ZREF@titleref@hook</code> . . . . .	949, 953, 957, 980	<code>\ZREF@xr@processfile</code> . . . . .	1138, <a href="#">1169</a> , 1197
<code>\zref@titleref@setcurrent</code> . . . . .	963, 1011, 1017, 1023, 1029, 1035, 1041, 1047, 1054, 1060, 1066, 1074, 1083, 1092	<code>\ZREF@xr@processline</code> . . . . .	1171, <a href="#">1177</a>
<code>\zref@titleref@stripperperiodtrue</code> . . . . .	962	<code>\ZREF@xr@refname</code> . . . . .	1201, 1204, 1215, 1223, 1226, 1228, 1237
<code>\ZREF@unexpanded</code> . . . . .	540, 542, 551, 553, 555	<code>\ZREF@xr@scanparams</code> . . . . .	1227, <a href="#">1271</a>
<code>\ZREF@UpdatePdfTeX</code> . . . . .	225, 1358, 1371	<code>\ZREF@xr@scantitleref</code> . . . . .	1274, <a href="#">1308</a>
<code>\ZREF@wrapper@babel</code> . . . . .	573, 579	<code>\ZREF@xr@url</code> . . . . .	1134, 1324
<code>\zref@wrapper@babel</code> . . . . .	8, 140, 491, 498, 556, 620, 625, 704, 999	<code>\ZREF@xr@urlcheck</code> . . . . .	1217, 1239, <a href="#">1318</a>
<code>\zref@wrapper@immediate</code> . . . . .	8, 89, <a href="#">439</a> , 689, 723	<code>\ZREF@xr@zref@newlabel</code> . . . . .	1181, 1244
<code>\zref@wrapper@unexpanded</code> . . . . .	8, <a href="#">550</a>	<code>\ZREF@xr@zreflabelfalse</code> . . . . .	1119
<code>\ZREF@X</code> . . . . .	378, 381, 392	<code>\ZREF@xr@zreflabeltrue</code> . . . . .	1122
<code>\ZREF@xr@input</code> . . . . .	1189, 1246	<code>\ZREF@zref</code> . . . . .	625, 628
<code>\ZREF@xr@checkfile</code> . . . . .	1135, 1138, 1166	<code>\zrefused</code> . . . . .	9, 94, 95, 161, 162, 163, 635
<code>\ZREF@xr@checkkey</code> . . . . .	1256, 1266	<code>\zruns</code> . . . . .	13, <a href="#">850</a> , 1480, 1485, 1491
<code>\ZREF@xr@checklist</code> . . . . .	1213, <a href="#">1254</a>	<code>\zsavapos</code> . . . . .	15, 157, 158, <a href="#">1381</a>
<code>\zref@xr@ext</code> . . . . .	16, <a href="#">1110</a> , 1131, 1331	<code>\zthepage</code> . . . . .	11, <a href="#">738</a>
<code>\ZREF@xr@externaldocument</code> . . . . .	1120, 1123, <a href="#">1126</a>	<code>\ztitleref</code> . . . . .	14, <a href="#">998</a>
		<code>\ztitlerefsetup</code> . . . . .	14, <a href="#">983</a>
		<code>\ztotpages</code> . . . . .	13, 125, <a href="#">839</a>
		<code>\zunknownnextpagename</code> . . . . .	12, 773, 816
		<code>\zunmakeperpage</code> . . . . .	14, <a href="#">933</a>
		<code>\zxrsetup</code> . . . . .	16, <a href="#">1330</a>