

The zref package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2010/03/29 v2.9

Abstract

Package `zref` tries to get rid of the restriction in \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

1	Introduction	3
1.1	Standard \LaTeX behaviour	3
1.2	Basic idea	4
1.3	Interfaces	4
2	Interface for programmers	4
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
3	User interface	9
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>marks</code>	13
3.8	Module <code>runs</code>	13
3.9	Module <code>perpage</code>	13
3.10	Module <code>counter</code>	14
3.11	Module <code>titleref</code>	14
3.12	Module <code>savepos</code>	15
3.13	Module <code>dotfill</code>	15
3.14	Module <code>xr</code>	16

4	ToDo	16
5	Example	16
6	Implementation	19
6.1	Package zref	19
6.1.1	Identification	19
6.1.2	Load basic module	19
6.1.3	Process options	19
6.2	Module base	19
6.2.1	Prefixes	19
6.2.2	Identification	20
6.2.3	Utilities	20
6.2.4	Check for ε -T _E X	20
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 babel	28
6.2.11	Unique counter support	28
6.2.12	Utilities	28
6.2.13	Setup	29
6.3	Module user	29
6.4	Module abspage	30
6.5	Module counter	31
6.6	Module lastpage	31
6.7	Module thepage	32
6.8	Module nextpage	33
6.9	Module totpages	34
6.10	Module marks	34
6.11	Module runs	36
6.12	Module perpage	36
6.13	Module titleref	38
6.13.1	Implementation	38
6.13.2	User interface	40
6.13.3	Patches for section and caption commands	40
6.14	Module xr	42
6.15	Module hyperref	46
6.16	Module savepos	47
6.16.1	Identification	47
6.16.2	Availability	47
6.16.3	Setup	47
6.16.4	User macros	48
6.17	Module dotfill	48
7	Test	49
7.1	\zref@localaddprop	49
7.2	Module runs	50
8	Installation	50
8.1	Download	50
8.2	Bundle installation	50
8.3	Package installation	51
8.4	Refresh file name databases	51
8.5	Some details for the interested	51
9	References	52

10 History	52
[2006/02/20 v1.0]	52
[2006/05/03 v1.1]	53
[2006/05/25 v1.2]	53
[2006/09/08 v1.3]	53
[2007/01/23 v1.4]	53
[2007/02/18 v1.5]	53
[2007/04/06 v1.6]	53
[2007/04/17 v1.7]	53
[2007/04/22 v1.8]	53
[2007/05/02 v1.9]	53
[2007/05/06 v2.0]	53
[2007/05/28 v2.1]	53
[2008/09/21 v2.2]	54
[2008/10/01 v2.3]	54
[2009/08/07 v2.4]	54
[2009/12/06 v2.5]	54
[2009/12/07 v2.6]	54
[2009/12/08 v2.7]	54
[2010/03/26 v2.8]	54
[2010/03/29 v2.9]	54
11 Index	54

1 Introduction

Standard L^AT_EX'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^AT_EX 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^AT_EX behaviour

References are created by the `\label` command:

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

Now L^AT_EX 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^AT_EX 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^AT_EX and other packages.

titleref stores its title data into the first entry in the list. L^AT_EX 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^AT_EX3 that will need some time before its first release. Thus I have implemented it as L^AT_EX 2_ε package without disturbing the existing L^AT_EX 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

^{exp} 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^AT_EX's `\currentlabel`.

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

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

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

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

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

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

2.4 Reference generation

`\zref@label {<refname>}`

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

`\zref@labelbylist {<refname>} {<listname>}`

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

`\zref@labelbyprops {<refname>} {<propnameA>,<propnameB>,...}`

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

`\zref@newlabel {<refname>} {...}`

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

2.5 Data extraction

`\zref@extractdefaultexp {<refname>} {<propname>} {<default>}`

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

`\zref@extractexp {<refname>} {<propname>}`

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

Example for page references:

```

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

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

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

`\zref@refused {⟨refname⟩}`

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

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

A possibility to check whether a reference exists.

`\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><empty></code>
	page	main	<code><empty></code>
abspage, totpages	abspage	main	0
perpage	pagevalue	perpage	0
	page	perpage	<code><empty></code>
	abspage	perpage	0
counter	counter	main	<code><empty></code>
titleref	title	main	<code><empty></code>
savepos	posx	savepos	0
	posy	savepos	0
hyperref	anchor	main	<code><empty></code>
	url		<code><empty></code>
xr	url		<code><empty></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 ε -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">• robust, unless the extracted values are fragile,• no babel shorthand support.

The basic user interface of the package without modules are commands that mimic the standard L^AT_EX 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`:

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

`\zpageref {⟨refname⟩}^babel`

Convenience macro, similar to `\pageref`.

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

`\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 \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}[2010/03/29]
44 \znextpagesetup
45   {\thepage}% next page is unknown
46   {\thepage\ (#1)}% this page is last page
47   {\thepage\ $\rightarrow$ #1}% next page is known
48 \renewcommand*{\znonextpagename}{last page}
49
50 \usepackage{fancyhdr}
51 \pagestyle{fancy}
52 \fancyhf{}
53 \fancyhead[LE,RO]{\znextpage}
54 \fancypagestyle{plain}{%
55   \fancyhf{}%
```

```

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

```

3.6 Module `totpages`

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

`\ztotpagesexp`

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

3.7 Module `marks`

ToDo.

3.8 Module `runs`

Module `runs` counts the \LaTeX runs since last `.aux` file creation and prints the number in the `.log` file.

`\zrunsexp`

Prints the total number of \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. \LaTeX runs where the `.aux` files are not rewritten are not counted (see `\nofiles`).

3.9 Module `perpage`

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

`\zmakeperpage [reset] {counter}`

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

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

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

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

```
\thezpage
counter zpage
```

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

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

```
\zunmakeperpage {⟨counter⟩}
```

The reset mechanism for this counter is deactivated.

3.10 Module counter

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

3.11 Module titleref

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

```
\ztitleref {⟨refname⟩}^babel
```

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

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

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

`title=⟨value⟩`

Sets the current title.

`stripperperiod=true|false`

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

`expand=true|false`

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

`cleanup={...}`

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

3.12 Module **savepos**

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

`\zsavepos {⟨refname⟩}`

It generates a reference with name $\langle refname \rangle$ to the location where the command is executed.

`\zposxexp {⟨refname⟩}`
`\zposyexp {⟨refname⟩}`

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

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

Both `\zposx` and `\zposy` are expandable and can be used inside calculations (`\setcounter`, `\addtocounter`, package `calc`, `\numexpr`). However this property prevents from notifying L^AT_EX that the reference is actually used (the notifying is not expandable). Therefore you should mark the reference as used by `\zrefused`.

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

3.13 Module **dotfill**

`\zdotfill`

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

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

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

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

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

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

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

`dot`= $\langle value \rangle$

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

3.14 Module **xr**

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

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

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

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

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

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

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

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

`\zref@xr@ext`

If the `<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 ist stored in references and are available for calculations in the next L^AT_EX 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^AT_EX is notified, if the references are not yet available and L^AT_EX 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 \dfptest{0.89em}
180 \dfptest{1.31em}
181 \dfptest{1.32em}
182 \dfptest{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 [2010/03/29 v2.9 New reference scheme for LaTeX2e (H0)]%

```

6.1.2 Load basic module

```

191 \RequirePackage{zref-base}[2010/03/29]

```

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}[2010/03/29]%
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 $\langle propname \rangle$ is expanded immediately or at shipout time.

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

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

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

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

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

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

6.2.2 Identification

```

213 \*base\
214 \NeedsTeXFormat{LaTeX2e}
215 \ProvidesPackage{zref-base}%
216 [2010/03/29 v2.9 Module base for zref (H0)]%
```

6.2.3 Utilities

```

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

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

```

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

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

```

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

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

```

220 \newif\ifZREF@found
```

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

```

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

6.2.4 Check for ε -TeX

The use of ε -TeX should be standard nowadays for L^AT_EX. We test for ε -TeX in order to use its features later.

```

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

```

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

```

6.2.5 Auxiliary file stuff

We are using some commands in the .aux files. However sometimes these auxiliary files are interpreted by L^AT_EX processes that haven't loaded this package (e.g. package xr). Therefore we provide dummy definitions.

```

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

```

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

- `\Z@R@labelname` is defined.
- L^AT_EX's check for multiple references.
- L^AT_EX's check for changed references.

```

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

```

6.2.6 Property lists

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

```

260 \def\zref@newlist#1{%
261   \zref@iflistundefined{#1}{%
262     \@ifdefinable{Z@L@#1}{%
263       \global\expandafter\let\csname Z@L@#1\endcsname\@empty
264       \PackageInfo{zref}{New property list: #1}%
265     }%
266   }{%
267     \PackageError\ZREF@name{%
268       Property list '#1' already exists%
269     }\@ehc
270   }%
271 }

```

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

```

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

```

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

```

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

```

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

```

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

```

`\zref@listforloop`

```

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

```

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

```

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

```

```

331 }

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

```

`\ZREF@l@addto@macro`

```

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

```

6.2.7 Properties

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

```

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

```

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

```

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

```

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

6.2.8 Reference generation

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

```

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

```

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

```

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

```

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

```

414 \def\zref@labelbyprops#1#2{%
415   \@bsphack
416   \begingroup

```



```

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

```

`\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.

```

434 \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`.

```

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

```

`\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.

```

443 \def\ZREF@label#1#2{%
444     \if@filesw
445         \begingroup
446         \ifZREF@immediate
447             \let\ZREF@org@thepage\thepage
448             \fi
449         \protected@write\@auxout{%
450             \ifZREF@immediate
451                 \let\thepage\ZREF@org@thepage
452             \fi
453             \let\ZREF@temp\@empty
454             \@tfor\ZREF@P:=#1\do{%
455                 \expandafter\ifx
456                     \csname\ifZREF@immediate relax\else Z@X@\ZREF@P\fi\endcsname
457                     \noexpand
458                     \expandafter\let\csname Z@C@\ZREF@P\endcsname\relax
459                 \fi
460             \toks@\expandafter{\ZREF@temp}%
461             \edef\ZREF@temp{%
462                 \the\toks@
463                 \expandafter\string\csname\ZREF@P\endcsname{%
464                     \expandafter\noexpand\csname Z@C@\ZREF@P\endcsname
465                 }%

```

```

466         }%
467     }%
468 }{%
469     \string\zref@newlabel{#2}{\ZREF@temp}%
470 }%
471 \endgroup
472 \fi
473 }
474 \def\ZREF@addtoks#1{%
475     \toks@\expandafter\expandafter\expandafter{%
476         \expandafter\the\expandafter\toks@#1%
477     }%
478 }

```

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^AT_EX 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 pdfL^AT_EX 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.

```

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

```

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

```

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

```

`\ZREF@ifrefundefined`

```

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

```

`\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^AT_EX and a warning is given.

```

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

```

`\ZREF@refused`

```

496 \def\ZREF@refused#1{%
497     \zref@ifrefundefined{#1}%

```

```

498 \protect\G@refundefinedtrue
499 \@latex@warning{%
500 Reference ‘#1’ on page \thepage \space undefined%
501 }%
502 }{}%
503 }

```

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

```

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

```

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

```

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

```

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

```

`\zref@wrapper@unexpanded`

```

546 \long\def\zref@wrapper@unexpanded#1{%
547 \let\ZREF@unexpanded\etex@unexpanded
548 #1%

```

```

549 \let\ZREF@unexpanded\ltx@firstofone
550 }
551 \let\ZREF@unexpanded\ltx@firstofone

```

6.2.10 Compatibility with babel

`\zref@wrapper@babel`

```

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

```

6.2.11 Unique counter support

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

```

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

```

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

```

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

```

6.2.12 Utilities

`\ZREF@number`

```

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

```

6.2.13 Setup

`\zref@setdefault` Standard L^AT_EX 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`.

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

`\zref@default` Now we initialize `\zref@default` with the same value that L^AT_EX uses for its undefined references.

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

Main property list.

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

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

Now we create the list.

```
604 \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^AT_EX 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

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

Mark successful loading

```
609 \let\ZREF@baseok\@empty  
610 </base>
```

6.3 Module user

```
611 <*user>  
612 \NeedsTeXFormat{LaTeX2e}  
613 \ProvidesPackage{zref-user}%  
614 [2010/03/29 v2.9 Module user for zref (HO)]%  
615 \RequirePackage{zref-base}[2010/03/29]  
616 \@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^AT_EX 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.

```

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

```

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

```

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

```

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

```

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

```

`\zrefused` For the following expandible user macros `\zrefused` should be used to notify L^AT_EX in case of undefined references.

```

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

```

6.4 Module `abspage`

```

638 <*abspage>
639 \NeedsTeXFormat{LaTeX2e}
640 \ProvidesPackage{zref-abspage}%
641 [2010/03/29 v2.9 Module abspage for zref (HO)]%
642 \RequirePackage{zref-base}[2010/03/29]
643 \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`.

```

644 \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_EX files.

```

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

```

655 </abspage>

```

6.5 Module counter

```

656 <*counter>
657 \NeedsTeXFormat{LaTeX2e}
658 \ProvidesPackage{zref-counter}%
659 [2010/03/29 v2.9 Module counter for zref (HO)]%
660 \RequirePackage{zref-base}[2010/03/29]
661 \@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.

```

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

```

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

```

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

```

6.6 Module lastpage

```

673 <*lastpage>
674 \NeedsTeXFormat{LaTeX2e}
675 \ProvidesPackage{zref-lastpage}%
676 [2010/03/29 v2.9 Module lastpage for zref (HO)]%
677 \RequirePackage{zref-base}[2010/03/29]
678 \RequirePackage{zref-abspage}[2010/03/29]
679 \RequirePackage{atveryend}[2009/12/07]
680 \@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.

```

681 \zref@newlist{LastPage}
682 \AfterLastShipout{%
683   \if@filesw
684     \begingroup
685       \advance\c@page\m@ne
686       \toks@{\expandafter\expandafter\expandafter{%
687         \expandafter\Z@L@main
688         \Z@L@LastPage
689       }}%
690       \expandafter\zref@wrapper@immediate\expandafter{%
691         \expandafter\ZREF@label\expandafter{\the\toks@}{LastPage}%
692       }%
693     \endgroup
694   \fi
695 }

```

`\zref@iflastpage`

```

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

```

```

703 }

\ziflastpage

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

ZREF@iflastpage

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

712 </lastpage>

```

6.7 Module thepage

```

713 <*thepage>
714 \NeedsTeXFormat{LaTeX2e}
715 \ProvidesPackage{zref-thepage}%
716   [2010/03/29 v2.9 Module thepage for zref (HO)]%
717 \RequirePackage{zref-base}[2010/03/29]
718 \ifundefined{ZREF@baseok}{\endinput}{}

719 \RequirePackage{atbegshi}
720 \RequirePackage{zref-abspage}[2010/03/29]

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

\zref@thepage@name

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

\zref@thepage

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

\zref@thepage@refused

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

\zthepage

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

743 </thepage>

```


6.8 Module nextpage

```

744 <*nextpage>
745 \NeedsTeXFormat{LaTeX2e}
746 \ProvidesPackage{zref-nextpage}%
747 [2010/03/29 v2.9 Module nextpage for zref (HO)]%
748 \RequirePackage{zref-base}[2010/03/29]
749 \@ifundefined{ZREF@baseok}{\endinput}{}

750 \RequirePackage{zref-abspage}[2010/03/29]
751 \RequirePackage{zref-thepage}[2010/03/29]
752 \RequirePackage{zref-lastpage}[2010/03/29]
753 \RequirePackage{uniquecounter}[2009/12/18]

754 \UniqueCounterNew{znextpage}
755
756 \newcommand*{\znextpagesetup}{%
757   \afterassignment\ZREF@np@setup@i
758   \def\ZREF@np@call@unknown##1%
759 }
760 \def\ZREF@np@setup@i{%
761   \afterassignment\ZREF@np@setup@ii
762   \def\ZREF@np@call@nonext##1%
763 }
764 \def\ZREF@np@setup@ii{%
765   \def\ZREF@np@call@next##1%
766 }
767 \def\ZREF@np@call@unknown#1{#1}
768 \def\ZREF@np@call@nonext#1{#1}
769 \def\ZREF@np@call@next#1{#1}
770 \newcommand*{\znextpage}{%
771   \UniqueCounterCall{znextpage}{\ZREF@nextpage}%
772 }
773 \newcommand*{\znonextpagename}{}
774 \newcommand*{\zunknownnextpagename}{\Z@D@page}
775 \def\ZREF@nextpage#1{%
776   \begingroup
777     \def\ZREF@refname@this{zref@np#1}%
778     \zref@labelbyprops\ZREF@refname@this{abspage}%
779     \chardef\ZREF@call=0 % unknown
780     \ZREF@ifrefundefined\ZREF@refname@this{%
781       }{%
782         \edef\ZREF@pagenum@this{%
783           \zref@extractdefault\ZREF@refname@this{abspage}{0}%
784         }%
785         \edef\ZREF@refname@next{%
786           \zref@thepage@name{%
787             \the\numexpr\ZREF@pagenum@this+1%
788           }%
789         }%
790         \ifnum\ZREF@pagenum@this>0 %
791           \ZREF@ifrefundefined{LastPage}{%
792             \zref@ifrefundefined\ZREF@refname@next{%
793               }{%
794                 \chardef\ZREF@call=2 % next page
795               }%
796             }{%
797               \edef\ZREF@pagenum@last{%
798                 \zref@extractdefault{LastPage}{abspage}{0}%
799               }%
800               \ifnum\ZREF@pagenum@this<\ZREF@pagenum@last\ltx@space
801                 \ZREF@ifrefundefined\ZREF@refname@next{%
802                   }{%
803                     \chardef\ZREF@call=2 % next page

```

```

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

```

6.9 Module totpages

```

832 <*totpages>
833 \NeedsTeXFormat{LaTeX2e}
834 \ProvidesPackage{zref-totpages}%
835 [2010/03/29 v2.9 Module totpages for zref (H0)]%
836 \RequirePackage{zref-base}[2010/03/29]
837 \@ifundefined{ZREF@baseok}{\endinput}{}

```

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

```

838 \RequirePackage{zref-abspage}[2010/03/29]
839 \RequirePackage{zref-lastpage}[2010/03/29]

```

\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.

```

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

```

Also we mark the reference `LastPage` as used:

```

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

```

6.10 Module marks

```

847 <*marks>
848 \NeedsTeXFormat{LaTeX2e}
849 \ProvidesPackage{zref-marks}%
850 [2010/03/29 v2.9 Module marks for zref (H0)]%
851 \RequirePackage{zref-base}[2010/03/29]
852 \@ifundefined{ZREF@baseok}{\endinput}{}
853 \RequirePackage{kvsetkeys}[2009/07/30]

```

```

854 \newcommand*{\zref@marks@register}[3][]{%
855   \edef\ZREF@TempName{#1}%
856   \edef\ZREF@TempNum{\ZREF@number#2}%
857   \ifnum\ZREF@TempNum<\ltx@zero %
858     \PackageError\ZREF@name{%
859       \string\zref@marks@register\ltx@space is called with invalid%
860       \MessageBreak
861       marks register number (\ZREF@TempNum)%
862     }{%
863       Use '0' or the command, defined by \string\newmarks.\MessageBreak
864       \@ehc
865     }%
866   \else
867     \ifx\ZREF@TempName\ltx@empty
868       \edef\ZREF@TempName{mark\romannumeral\ZREF@TempNum}%
869     \else
870       \edef\ZREF@TempName{marks\ZREF@TempName}%
871     \fi
872     \ZREF@MARKS@DefineProp{top}%
873     \ZREF@MARKS@DefineProp{first}%
874     \ZREF@MARKS@DefineProp{bot}%
875     \kv@parse{#3}{%
876       \ifx\kv@value\relax
877         \def\kv@value{top,first,bot}%
878       \fi
879       \edef\ZREF@temp{\expandafter\@car\kv@key X\@nil}%
880       \ifx\ZREF@temp\ZREF@STAR
881         \edef\kv@key{\expandafter\@cdr\kv@key\@nil}%
882         \zref@newlist\kv@key
883       \fi
884       \expandafter\comma@parse\expandafter{\kv@value}{%
885         \ifcase0\ifx\comma@entry\ZREF@NAME@top 1\else
886           \ifx\comma@entry\ZREF@NAME@first 1\else
887             \ifx\comma@entry\ZREF@NAME@bot 1\fi\fi\fi\ltx@space
888         \PackageWarning\ZREF@name{%
889           Use 'top', 'first' or 'bot' for the list values%
890           \MessageBreak
891           in the third argument of \string\zref@marks@register.%
892           \MessageBreak
893           Ignoring unkown value '\comma@entry'%
894         }%
895       \else
896         \zref@addprop{\kv@key}{\comma@entry\ZREF@TempName}%
897       \fi
898       \ltx@gobble
899     }%
900     \ltx@gobbletwo
901   }%
902 \fi
903 }
904 \def\ZREF@STAR{*}
905 \def\ZREF@NAME@top{top}
906 \def\ZREF@NAME@first{first}
907 \def\ZREF@NAME@bot{bot}
908 \def\ZREF@MARKS@DefineProp#1{%
909   \zref@ifpropundefined{#1\ZREF@TempName}{%
910     \ifnum\ZREF@TempNum=\ltx@zero
911       \begingroup
912       \edef\x{\endgroup
913         \noexpand\zref@newprop*{#1\ZREF@TempName}[]{%
914           \expandafter\noexpand\csname#1mark\endcsname
915         }%

```

```

916     }%
917     \x
918     \else
919     \begingroup
920     \edef\x{\endgroup
921       \noexpand\zref@newprop*{#1\ZREF@TempName}[]{%
922         \expandafter\noexpand\csname#1marks\endcsname
923         \ZREF@TempNum
924       }%
925     }%
926     \x
927     \fi
928   }{%
929     \PackageWarning\ZREF@name{%
930       \string\zref@marks@register\ltx@space does not generate the%
931       \MessageBreak
932       new property ‘#1\ZREF@TempName’, because\MessageBreak
933       it is already defined%
934     }%
935   }%
936 }
937 </marks>

```

6.11 Module runs

This module does not use the label-reference-system. The reference changes with each L^AT_EX run and would force a rerun warning always.

```

938 <*runs>
939 \NeedsTeXFormat{LaTeX2e}
940 \ProvidesPackage{zref-runs}%
941 [2010/03/29 v2.9 Module runs for zref (H0)]%

```

\zruns

```

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

```

6.12 Module perpage

```

956 <*perpage>
957 \NeedsTeXFormat{LaTeX2e}
958 \ProvidesPackage{zref-perpage}%
959 [2010/03/29 v2.9 Module perpage for zref (H0)]%
960 \RequirePackage{zref-base}[2010/03/29]
961 \@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.

```

962 \RequirePackage{zref-abspage}[2010/03/29]

```

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

```

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

```

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

```

968 \newcounter{zpage}

```

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

```

969 \zref@require@unique

```

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

```

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

```

`\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.

```

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

```

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

```

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

987 \def\ZREF@@makeperpage[#1]#2{%
988   \@ifundefined{@stepcounterhook@#2}{%
989     \expandafter\gdef\csname @stepcounterhook@#2\endcsname{%
990     }%
991     \expandafter\gdef\csname ZREF@perpage@#2\endcsname{%
992       \ZREF@perpage@step{#2}{#1}%
993     }%
994     \expandafter\g@addto@macro\csname @stepcounterhook@#2\endcsname{%
995       \ifcsname ZREF@perpage@#2\endcsname
996         \csname ZREF@perpage@#2\endcsname
997       \fi
998     }%
999 }

```

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

```

1000 \def\ZREF@@perpage@step#1#2{%

```

First the reference is generated.

```

1001   \global\advance\c@zref@unique\@ne
1002   \begingroup
1003     \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.

```

1004 \zref@ifrefundefined\thezref@unique{%
1005   \global\c@zpage=\c@page
1006   \global\let\thezpage\thepage
1007   \expandafter\xdef\csname ZREF@abspage@#1\endcsname{\number\c@abspage}%
1008 }%
```

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

```

1009 \global\c@zpage=\zref@extract\thezref@unique{pagevalue}\relax
1010 \xdef\thezpage{\noexpand\zref@extract{\thezref@unique}{page}}%
1011 \expandafter\xdef\csname ZREF@abspage@#1\endcsname{%
1012   \zref@extractdefault\thezref@unique{abspage}{\number\c@abspage}%
1013 }%
1014 }%
```

Page changes are detected by a changed absolute page number.

```

1015 \expandafter\ifx\csname ZREF@abspage@#1\expandafter\endcsname
1016   \csname ZREF@currentabspage@#1\endcsname
1017 \else
1018   \global\csname c@#1\endcsname=#2\relax
1019   \global\expandafter\let
1020     \csname ZREF@currentabspage@#1\expandafter\endcsname
1021     \csname ZREF@abspage@#1\endcsname
1022 \fi
1023 \endgroup
1024 }
```

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

```

1025 \newcommand*{\zunmakeperpage}[1]{%
1026   \global\expandafter\let\csname ZREF@perpage@#1\endcsname\@undefined
1027 }
1028 </perpage>
```

6.13 Module `titleref`

```

1029 <*titleref>
1030 \NeedsTeXFormat{LaTeX2e}
1031 \ProvidesPackage{zref-titleref}%
1032 [2010/03/29 v2.9 Module titleref for zref (HO)]%
1033 \RequirePackage{zref-base}[2010/03/29]
1034 \@ifundefined{ZREF@baseok}{\endinput}{}
1035 \RequirePackage{getttitlestring}[2009/12/08]
```

6.13.1 Implementation

```
1036 \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`.

```
1037 \let\zref@titleref@current\@empty
```

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

```

1038 \zref@newprop{title}{\zref@titleref@current}%
1039 \zref@addprop{ZREF@mainlist}{title}%
```

The title strings go into the `.aux` file, thus they need some kind of protection. Package `titleref` uses a protected expansion method. The advantage is that this

can be used to cleanup the string and to remove `\label`, `\index` and other macros unwanted for referencing. But there is the risk that fragile stuff can break.

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

<code>\ifzref@titleref@expand</code>	<p>The switch <code>\ifzref@titleref@expand</code> distinguishes between the both methods. Package <code>nameref</code>'s behaviour is achieved by setting the switch to false, otherwise <code>titleref</code>'s expansion is used. Default is false.</p> <pre>1040 \newif\ifzref@titleref@expand</pre>
<code>\ZREF@titleref@hook</code>	<p>The hook <code>\ZREF@titleref@hook</code> allows to extend the cleanup for the expansion method. Thus unnecessary macros can be removed or dangerous commands removed. The hook is executed before the expansion of <code>\zref@titleref@current</code>.</p> <pre>1041 \let\ZREF@titleref@hook\@empty</pre>
<code>\zref@titleref@cleanup</code>	<p>The hook should not be used directly, instead we provide the macro <code>\zref@titleref@cleanup</code> to add stuff to the hook and prevents that a previous non-empty content is not discarded accidentally.</p> <pre>1042 \def\zref@titleref@cleanup#1{% 1043 \begingroup 1044 \toks@{\expandafter{% 1045 \ZREF@titleref@hook 1046 #1% 1047 }}% 1048 \expandafter\endgroup 1049 \expandafter\def\expandafter\ZREF@titleref@hook\expandafter{% 1050 \the\toks@ 1051 }% 1052 }%</pre>
<code>\ifzref@titleref@stripperperiod</code>	<p>Sometimes a title contains a period at the end. Package <code>nameref</code> removes this. This behaviour is controlled by the switch <code>\ifzref@titleref@stripperperiod</code> and works regardless of the setting of option <code>expand</code>. Period stripping is the default.</p> <pre>1053 \newif\ifzref@titleref@stripperperiod 1054 \zref@titleref@stripperperiodtrue</pre>
<code>\zref@titleref@setcurrent</code>	<p>Macro <code>\zref@titleref@setcurrent</code> sets a new current title stored in <code>\zref@titleref@current</code>. Some cleanup and expansion is performed that can be controlled by the previous switches.</p> <pre>1055 \def\zref@titleref@setcurrent#1{% 1056 \ifzref@titleref@expand 1057 \GetTitleStringExpand{#1}% 1058 \else 1059 \GetTitleStringNonExpand{#1}% 1060 \fi 1061 \edef\zref@titleref@current{% 1062 \detokenize\expandafter{\GetTitleStringResult}% 1063 }% 1064 \ifzref@titleref@stripperperiod 1065 \edef\zref@titleref@current{% 1066 \expandafter\ZREF@stripperperiod\zref@titleref@current 1067 \@empty.\@empty\@nil 1068 }% 1069 \fi 1070 }% 1071 \GetTitleStringDisableCommands{% 1072 \ZREF@titleref@hook 1073 }</pre>

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

```
1074 \def\ZREF@stripperperiod#1.\@empty#2\@nil{#1}%
```

6.13.2 User interface

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

```
1075 \define@key{ZREF@TR}{expand}[true]{%
1076   \csname zref@titleref@expand#1\endcsname
1077 }%
1078 \define@key{ZREF@TR}{stripperperiod}[true]{%
1079   \csname zref@titleref@stripperperiod#1\endcsname
1080 }%
1081 \define@key{ZREF@TR}{cleanup}{%
1082   \zref@titleref@cleanup{#1}%
1083 }%
1084 \define@key{ZREF@TR}{title}{%
1085   \def\zref@titleref@current{#1}%
1086 }%
1087 \newcommand*{\ztitlerefsetup}{%
1088   \setkeys{ZREF@TR}%
1089 }%
```

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

```
1090 \newcommand*{\ztitleref}{%
1091   \zref@wrapper@babel\ZREF@titleref
1092 }%
1093 \def\ZREF@titleref#1{%
1094   \begingroup
1095     \zref@refused{#1}%
1096     \let\label\ltx@gobble
1097     \zref@extract{#1}{title}%
1098   \endgroup
1099 }%
```

6.13.3 Patches for section and caption commands

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

Captions of figures and tables.

```
1100 \AtBeginDocument{%
1101   \ZREF@patch{@caption}{%
1102     \long\def\@caption#1[#2]{%
1103       \zref@titleref@setcurrent{#2}%
1104       \ZREF@org@caption{#1}[{#2}]%
1105     }%
1106   }%
```

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

```
1107   \ZREF@patch{@part}{%
1108     \def\@part[#1]{%
1109       \zref@titleref@setcurrent{#1}%
1110       \ZREF@org@part[{#1}]%
1111     }%
1112   }%
1113   \ZREF@patch{@chapter}{%
1114     \def\@chapter[#1]{%
1115       \zref@titleref@setcurrent{#1}%
```



```

1116     \ZREF@org@@chapter[{#1}]%
1117 }%
1118 }%
1119 \ZREF@patch{@sect}{%
1120   \def\@sect#1#2#3#4#5#6[#7]{%
1121     \zref@titleref@setcurrent{#7}%
1122     \ZREF@org@@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}]%
1123   }%
1124 }%

```

The star versions of the section commands.

```

1125 \ZREF@patch{@spart}{%
1126   \def\@spart#1{%
1127     \zref@titleref@setcurrent{#1}%
1128     \ZREF@org@@spart{#1}%
1129   }%
1130 }%
1131 \ZREF@patch{@schapter}{%
1132   \def\@schapter#1{%
1133     \zref@titleref@setcurrent{#1}%
1134     \ZREF@org@@schapter{#1}%
1135   }%
1136 }%
1137 \ZREF@patch{@ssect}{%
1138   \def\@ssect#1#2#3#4#5{%
1139     \zref@titleref@setcurrent{#5}%
1140     \ZREF@org@@ssect{#1}{#2}{#3}{#4}{#5}%
1141   }%
1142 }%

```

Class beamer.

```

1143 \@ifclassloaded{beamer}{%
1144   \ZREF@patch{beamer@section}{%
1145     \long\def\beamer@section[#1]{%
1146       \zref@titleref@setcurrent{#1}%
1147       \ZREF@org@beamer@section[{#1}]%
1148     }%
1149   }%
1150   \ZREF@patch{beamer@subsection}{%
1151     \long\def\beamer@subsection[#1]{%
1152       \zref@titleref@setcurrent{#1}%
1153       \ZREF@org@beamer@subsection[{#1}]%
1154     }%
1155   }%
1156   \ZREF@patch{beamer@subsubsection}{%
1157     \long\def\beamer@subsubsection[#1]{%
1158       \zref@titleref@setcurrent{#1}%
1159       \ZREF@org@beamer@subsubsection[{#1}]%
1160     }%
1161   }%
1162 }{}%

```

Package titlesec.

```

1163 \@ifpackageloaded{titlesec}{%
1164   \ZREF@patch{ttl@sect@i}{%
1165     \def\ttl@sect@i#1#2[#3]#4{%
1166       \zref@titleref@setcurrent{#4}%
1167       \ZREF@org@ttl@sect@i{#1}{#2}[{#3}]{#4}%
1168     }%
1169   }%
1170 }{}%

```

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

```

1171 \ifpackageloaded{longtable}{%
1172   \ZREF@patch{LT@c@ption}{%
1173     \def\LT@c@ption#1[#2]#3{%
1174       \ZREF@org@LT@c@ption{#1}[{#2}]{#3}%
1175       \zref@titleref@setcurrent{#2}%
1176     }%
1177   }%
1178 }{}%

Package listings: support for its caption.

1179 \ifpackageloaded{listings}{%
1180   \ZREF@patch{lst@MakeCaption}{%
1181     \def\lst@MakeCaption{%
1182       \ifx\lst@label\@empty
1183       \else
1184         \expandafter\zref@titleref@setcurrent\expandafter{%
1185           \lst@@caption
1186         }%
1187       \fi
1188       \ZREF@org@lst@MakeCaption
1189     }%
1190   }%
1191 }{}%
1192 }
1193 </titleref>

```

6.14 Module xr

```

1194 < *xr>
1195 \NeedsTeXFormat{LaTeX2e}
1196 \ProvidesPackage{zref-xr}%
1197 [2010/03/29 v2.9 Module xr for zref (H0)]%
1198 \RequirePackage{zref-base}[2010/03/29]
1199 \ifundefined{ZREF@baseok}{\endinput}{}
1200 \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.

```
1201 \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.

```

1202 \providecommand*{\zref@xr@ext}{}%
1203 \ltx@ifundefined{XR@ext}{pdf}{\XR@ext}%
1204 }%

```

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

```
1205 \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).

```

1206 \newcommand*{\zexternaldocument}{%
1207   \begingroup
1208     \csname @safe@actives@true\endcsname
1209     \makeatletter
1210     \ifstar{%
1211       \ZREF@xr@zreflabelfalse
1212       \@testopt\ZREF@xr@externaldocument{}%

```

```

1213 }{%
1214   \ZREF@xr@zreflabeltrue
1215   \@testopt\ZREF@xr@externaldocument{}%
1216 }%
1217 }%

```

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

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

```

1218 \def\ZREF@xr@externaldocument[#1]#2{%
1219   \def\ZREF@xr@prefix{#1}%
1220   \let\ZREF@xr@filelist\@empty
1221   \edef\ZREF@xr@file{#2.aux}%
1222   \filename@parse{#2}%
1223   \@testopt\ZREF@xr@graburl{#2.\zref@xr@ext}%
1224 }%
1225 \def\ZREF@xr@graburl[#1]{%
1226   \edef\ZREF@xr@url{#1}%
1227   \ZREF@xr@checkfile
1228 \endgroup
1229 }%

```

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

```

1230 \def\ZREF@xr@checkfile{%
1231   \openin\@inputcheck\ZREF@xr@file\relax
1232   \ifeof\@inputcheck
1233     \PackageWarning{zref-xr}{%
1234       File '\ZREF@xr@file' not found or empty,\MessageBreak
1235       labels not imported%
1236     }%
1237   \else
1238     \PackageInfo{zref-xr}{%
1239       Label \ifZREF@xr@zreflabel (zref) \fi import from '\ZREF@xr@file'%
1240     }%
1241     \def\ZREF@xr@found{0}%
1242     \def\ZREF@xr@ignored{0}%
1243     \ZREF@xr@processfile
1244     \closein\@inputcheck
1245     \begingroup
1246       \let\on@line\@empty
1247       \PackageInfo{zref-xr}{%
1248         Statistics for '\ZREF@xr@file': %
1249         \ZREF@xr@found\space found, %
1250         \ZREF@xr@ignored\space ignored%
1251       }%
1252     \endgroup
1253   \fi
1254   \ifx\ZREF@xr@filelist\@empty
1255   \else
1256     \edef\ZREF@xr@file{\expandafter\@car\ZREF@xr@filelist\@nil}%
1257     \edef\ZREF@xr@filelist{\expandafter\@cdr\ZREF@xr@filelist\@nil}%
1258     \expandafter\ZREF@xr@checkfile
1259   \fi
1260 }%

```

`\ZREF@xr@processfile`

```

1261 \def\ZREF@xr@processfile{%

```

```

1262 \read\@inputcheck to\ZREF@xr@line
1263 \expandafter\ZREF@xr@processline\ZREF@xr@line..\ZREF@nil
1264 \ifeof\@inputcheck
1265 \else
1266 \expandafter\ZREF@xr@procesfile
1267 \fi
1268 }%

```

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

```

1269 \long\def\ZREF@xr@processline#1#2#3\ZREF@nil{%
1270 \def\x{#1}%
1271 \toks@{#2}%
1272 \ifZREF@xr@zreflabel
1273 \ifx\x\ZREF@xr@zref@newlabel
1274 \expandafter\ZREF@xr@process@zreflabel\ZREF@xr@line...\ZREF@nil
1275 \fi
1276 \else
1277 \ifx\x\ZREF@xr@newlabel
1278 \expandafter\ZREF@xr@process@label\ZREF@xr@line...[]\ZREF@nil
1279 \fi
1280 \fi
1281 \ifx\x\ZREF@xr@@input
1282 \edef\ZREF@xr@filelist{%
1283 \etex@unexpanded\expandafter{\ZREF@xr@filelist}%
1284 {\filename@area\the\toks@}%
1285 }%
1286 \fi
1287 \ifeof\@inputcheck
1288 \else
1289 \expandafter\ZREF@xr@processfile
1290 \fi
1291 }%
1292 \def\ZREF@xr@process@zreflabel\zref@newlabel#1#2#3\ZREF@nil{%
1293 \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1294 \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1295 \def\x{#2}%
1296 \@ifundefined{\ZREF@xr@refname}{%
1297 \let\ZREF@xr@list\x
1298 \ifx\ZREF@xr@list\@empty
1299 \PackageWarningNoLine{zref-xr}{%
1300 Label ‘#1’ without properties ignored\MessageBreak
1301 in file ‘\ZREF@xr@file’%
1302 }%
1303 \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
1304 \else
1305 \expandafter\ZREF@xr@checklist\x\ZREF@nil
1306 \expandafter\global\expandafter\let
1307 \csname \ZREF@xr@refname\endcsname\x
1308 \fi
1309 \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1310 }{%
1311 \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
1312 }%
1313 }%
1314 \def\ZREF@xr@process@label\newlabel#1#2#3[#4]#5\ZREF@nil{%
1315 \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
1316 \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
1317 \def\x{#2}%
1318 \@ifundefined{\ZREF@xr@refname}{%
1319 \expandafter\ZREF@xr@scanparams
1320 \csname\ZREF@xr@refname\expandafter\endcsname
1321 \x{}{}{}{}{}\ZREF@nil
1322 \ifx\\#4\\%

```

```

1323 \else
1324 % ntheorem knows an optional argument at the end of \newlabel
1325 \zref@ifpropundefined{theotype}{%
1326 \zref@newprop{theotype}{}%
1327 }{}%
1328 \expandafter\g@addto@macro
1329 \csname\ZREF@xr@refname\endcsname{\theotype{#4}}%
1330 \fi
1331 \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
1332 }{%
1333 \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
1334 }%
1335 }
1336 \def\ZREF@xr@zref@newlabel{\zref@newlabel}%
1337 \def\ZREF@xr@newlabel{\newlabel}%
1338 \def\ZREF@xr@input{\@input}%

```

\ZREF@xr@ignorewarning

```

1339 \def\ZREF@xr@ignorewarning#1{%
1340 \PackageWarningNoLine{zref-xr}{%
1341 Label ‘#1’ is already in use\MessageBreak
1342 in file ‘\ZREF@xr@file’%
1343 }%
1344 \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
1345 }%

```

\ZREF@xr@checklist

```

1346 \def\ZREF@xr@checklist#1#2#3\ZREF@nil{%
1347 \ifx\@undefined#1\relax
1348 \expandafter\ZREF@xr@checkkey|string#1\@nil
1349 \fi
1350 \ifx\#3\%
1351 \else
1352 \@ReturnAfterFi{%
1353 \ZREF@xr@checklist#3\ZREF@nil
1354 }%
1355 \fi
1356 }%
1357 \long\def\@ReturnAfterFi#1\fi{\fi#1}%
1358 \def\ZREF@xr@checkkey#1#2\@nil{%
1359 \zref@ifpropundefined{#2}{%
1360 \zref@newprop{#2}{}%
1361 }{}%
1362 }%

```

\ZREF@xr@scanparams

```

1363 \def\ZREF@xr@scanparams#1#2#3#4#5#6#7\ZREF@nil{%
1364 \global\let#1\@empty
1365 \ZREF@foundfalse
1366 \ZREF@xr@scantitleref#1#2\TR@TitleReference{}{}\ZREF@nil
1367 \ifZREF@found
1368 \else
1369 \g@addto@macro#1{\default{#2}}%
1370 \fi
1371 % page
1372 \g@addto@macro#1{\page{#3}}%
1373 % nameref title
1374 \ifZREF@found
1375 \else
1376 \ifx\#4\%
1377 \else
1378 \zref@ifpropundefined{title}{%

```

```

1379         \zref@newprop{title}{}%
1380     }{}%
1381     \g@addto@macro#1{\title{#4}}%
1382     \fi
1383 \fi
1384 % anchor
1385 \ifx\\#5\\%
1386 \else
1387     \zref@ifpropundefined{anchor}{%
1388         \zref@newprop{anchor}{}%
1389     }{}%
1390     \g@addto@macro#1{\anchor{#5}}%
1391 \fi
1392 \ifx\\#6\\%
1393 \else
1394     \zref@ifpropundefined{url}{%
1395         \zref@newprop{url}{}%
1396     }{}%
1397     \g@addto@macro#1{\url{#6}}%
1398 \fi
1399 }%

```

\ZREF@xr@scantitleref

```

1400 \def\ZREF@xr@scantitleref#1#2\TR@TitleReference#3#4#5\ZREF@nil{%
1401     \ifx\\#5\\%
1402     \else
1403         \g@addto@macro#1{%
1404             \default{#3}%
1405             \title{#4}%
1406         }%
1407         \ZREF@foundtrue
1408     \fi
1409 }%

```

\ZREF@xr@urlcheck

```

1410 \def\ZREF@xr@urlcheck#1{%
1411     \zref@ifrefcontainsprop{#1}{anchor}{%
1412         \zref@ifrefcontainsprop{#1}{url}{%
1413             {%
1414                 \expandafter\g@addto@macro\csname Z@R@#1\expandafter\endcsname
1415                 \expandafter{%
1416                     \expandafter\url\expandafter{\ZREF@xr@url}%
1417                 }%
1418             }%
1419         }{%
1420         }%
1421 }%

```

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

```

1422 \define@key{ZREF@XR}{ext}{%
1423     \def\zref@xr@ext{#1}%
1424 }%
1425 \newcommand*{\zxrsetup}{%
1426     \setkeys{ZREF@XR}%
1427 }%
1428 \</xr>

```

6.15 Module hyperref

UNFINISHED :-)

```

1429 <*hyperref>
1430 \NeedsTeXFormat{LaTeX2e}
1431 \ProvidesPackage{zref-hyperref}%
1432 [2010/03/29 v2.9 Module hyperref for zref (HO)]%
1433 \RequirePackage{zref-base}[2010/03/29]
1434 \@ifundefined{ZREF@baseok}{\endinput}{}
1435 \zref@newprop{anchor}[]{}%
1436 \ltx@ifundefined{@currentHref}{\@currentHref}%
1437 }%
1438 \zref@addprop{ZREF@mainlist}{anchor}%
1439 </hyperref>

```

6.16 Module **savepos**

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

6.16.1 Identification

```

1440 <*savepos>
1441 \NeedsTeXFormat{LaTeX2e}
1442 \ProvidesPackage{zref-savepos}%
1443 [2010/03/29 v2.9 Module savepos for zref (HO)]%
1444 \RequirePackage{zref-base}[2010/03/29]
1445 \@ifundefined{ZREF@baseok}{\endinput}{}

```

6.16.2 Availability

First we check, whether the feature is available.

```

1446 \ltx@ifundefined{pdfsavepos}{%
1447   \PackageError{ZREF@name}{%
1448     \string\pdfsavepos\space is not supported.\MessageBreak
1449     It is provided by pdfTeX (1.40) or XeTeX%
1450   }\ZREF@UpdatePdfTeX
1451   \endinput
1452 }{}%

```

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`.

```

1453 \RequirePackage{ifpdf}
1454 \ifpdf
1455 \else
1456   \begingroup\expandafter\expandafter\expandafter\endgroup
1457   \expandafter\ifx\csname pdftexversion\endcsname\relax
1458   \else
1459     \ifnum\pdftexversion<140 %
1460       \PackageError{ZREF@name}{%
1461         \string\pdfsavepos\space is not supported in DVI mode\MessageBreak
1462         of this pdfTeX version%
1463       }\ZREF@UpdatePdfTeX
1464       \expandafter\expandafter\expandafter\endinput
1465     \fi
1466   \fi
1467 \fi

```

6.16.3 Setup

```

1468 \zref@newlist{savepos}
1469 \zref@newprop*{posx}[0]{\the\pdflastxpos}
1470 \zref@newprop*{posy}[0]{\the\pdflastypos}
1471 \zref@addprop{savepos}{posx}

```

```
1472 \zref@addprop{savepos}{posy}
```

6.16.4 User macros

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

```
1473 \def\zsavepos#1{%
1474   \@bsphack
1475   \if@filesw
1476     \pdfsavepos
1477     \zref@labelbylist{#1}{savepos}%
1478   \fi
1479   \@esphack
1480 }
```

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

\zposy

```
1481 \newcommand*{\zposx}[1]{%
1482   \zref@extract{#1}{posx}%
1483 }%
1484 \newcommand*{\zposy}[1]{%
1485   \zref@extract{#1}{posy}%
1486 }%
```

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 mark used positions by **\zrefused** in order to notify L^AT_EX about undefined references.

```
1487 \</savepos>
```

6.17 Module dotfill

```
1488 <*dotfill>
1489 \NeedsTeXFormat{LaTeX2e}
1490 \ProvidesPackage{zref-dotfill}%
1491   [2010/03/29 v2.9 Module dotfill for zref (HO)]%
1492 \RequirePackage{zref-base}[2010/03/29]
1493 \ifundefined{ZREF@baseok}{\endinput}{}
```

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

```
1494 \RequirePackage{zref-savepos}[2010/03/29]
```

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

```
1495 \zref@require@unique
```

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

```
1496 \RequirePackage{keyval}
```

The definitions of the keys follow.

```
1497 \define@key{ZREF@DF}{unit}{%
1498   \def\ZREF@df@unit{#1}%
1499 }
1500 \define@key{ZREF@DF}{min}{%
1501   \def\ZREF@df@min{#1}%
1502 }
1503 \define@key{ZREF@DF}{dot}{%
1504   \def\ZREF@df@dot{#1}%
1505 }
```

Defaults are set, see user interface.

```
1506 \providecommand\ZREF@df@min{2}
1507 \providecommand\ZREF@df@unit{.44em}
```



```

1508 \providecommand\ZREF@df@dot{.}
\zdotfillsetup Configuration of \zdotfill is done by \zdotfillsetup.
1509 \newcommand*{\zdotfillsetup}{\setkeys{ZREF@DF}}

\zdotfill \zdotfill sets labels at the left and the right to get the horizontal position.
\zsavepos is not used, because we do not need the vertical position.
1510 \newcommand*{\zdotfill}{%
1511 \leavevmode
1512 \global\advance\c@zref@unique\@ne
1513 \begingroup
1514 \def\ZREF@temp{zref@\number\c@zref@unique}%
1515 \pdfsavepos
1516 \zref@labelbyprops{\thezref@unique L}{posx}%
1517 \setlength{\dimen@}{\ZREF@df@unit}%
1518 \zref@ifrefundefined{\thezref@unique R}{%
1519 \ZREF@dotfill
1520 }{%
1521 \ifnum\numexpr\zposx{\thezref@unique R}-\zposx{\thezref@unique L}\relax
1522 <\dimexpr\ZREF@df@min\dimen@\relax
1523 \hfill
1524 \else
1525 \ZREF@dotfill
1526 \fi
1527 }%
1528 \pdfsavepos
1529 \zref@labelbyprops{\thezref@unique R}{posx}%
1530 \endgroup
1531 \kern\z@
1532 }

\ZREF@dotfill Help macro that actually sets the dots.
1533 \def\ZREF@dotfill{%
1534 \cleaders\hb@xt@\dimen@{\hss\ZREF@df@dot\hss}\hfill
1535 }

1536 </dotfill>

```

7 Test

7.1 \zref@localaddprop

```

1537 <*test1>
1538 \NeedsTeXFormat{LaTeX2e}
1539 \nofiles
1540 \documentclass{article}
1541 \usepackage{zref-base}[2010/03/29]
1542 \usepackage{qstest}
1543 \IncludeTests{*}
1544 \LogTests{log}{*}{*}
1545
1546 \makeatletter
1547 \begin{qstest}{\localaddprop}{\localaddprop}
1548 \Expect*{\Z@L@main}{\{default\}{page}}%
1549 \zref@newprop{foobar}{F00}%
1550 \zref@newlist{alist}%
1551 \Expect*{\Z@L@alist}{\}%
1552 \begingroup
1553 \zref@localaddprop{main}{foobar}%
1554 \Expect*{\Z@L@main}{\{default\}{page}{foobar}}%
1555 \zref@localaddprop{alist}{page}%
1556 \Expect*{\Z@L@alist}{\{page}}%

```

```

1557 \endgroup
1558 \Expect*{\Z@L@main}*{\default}{page}}%
1559 \Expect*{\Z@L@alist}{}%
1560 \end{qstest}
1561 \@@end
1562 \</test1>

```

7.2 Module runs

```

1563 <*test-runs>
1564 \NeedsTeXFormat{LaTeX2e}
1565 \documentclass{article}
1566 \usepackage{zref-runs}[2010/03/29]
1567 \usepackage{qstest}
1568 \IncludeTests{*}
1569 \LogTests{log}{*}{*}
1570
1571 \begin{qstest}{zruns-preamble}{zruns-preamble}
1572 \Expect{0}*{\zruns}%
1573 \end{qstest}
1574
1575 \AtBeginDocument{%
1576 \begin{qstest}{zruns-atbegindocument}{zruns-atbegindocument}%
1577 \Expect*{\number\ExpectRuns}*{\zruns}%
1578 \end{qstest}%
1579 }
1580
1581 \begin{document}
1582 \begin{qstest}{zruns-document}{zruns-document}
1583 \Expect*{\number\ExpectRuns}*{\zruns}%
1584 \end{qstest}
1585 \end{document}
1586 \</test-runs>

```

8 Installation

8.1 Download

Package. This package is available on CTAN¹:

[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 \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
```

¹<ftp://ftp.ctan.org/tex-archive/>

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

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

8.3 Package installation

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

```
tex zref.dtx
```

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

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

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

8.4 Refresh file name databases

If your `TEX` distribution (te`TEX`, mik`TEX`, ...) relies on file name databases, you must refresh these. For example, te`TEX` users run `texhash` or `mktextlsr`.

8.5 Some details for the interested

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

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

Unpacking with L^AT_EX. The .dtx chooses its action depending on the format:

plain-T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for docstrip (really, docstrip does not need L^AT_EX), then inform the autodetect routine about your intention:

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

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

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

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

An example follows how to generate the documentation with pdfL^AT_EX:

```
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 ε -TeX's `\unexpanded`.

[2007/05/28 v2.1]

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

[2008/09/21 v2.2]

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

[2008/10/01 v2.3]

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

[2009/08/07 v2.4]

- Module `runs` added.

[2009/12/06 v2.5]

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

[2009/12/07 v2.6]

- Version date added for package `atveryend`.

[2009/12/08 v2.7]

- Module `titleref`: Use of package `getttitlestring`.

[2010/03/26 v2.8]

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

[2010/03/29 v2.9]

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

11 Index

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

Symbols	
<code>\@@end</code>	<i>1561</i>
<code>\@ReturnAfterFi</code>	<i>1352, 1357</i>
<code>\@addtoreset</code>	<i>581, 646</i>
<code>\@auxout</code>	<i>449</i>
<code>\@bsphack</code>	<i>405, 415, 1474</i>
<code>\@caption</code>	<i>1102</i>
<code>\@car</code>	<i>879, 1256</i>

<code>\@cdr</code>	881, 1257		
<code>\@chapter</code>	1114		
<code>\@currentHref</code>	1436		
<code>\@currentlabel</code>	605		
<code>\@ehc</code>	269, 283, 369, 864		
<code>\@empty</code>	263, 377, 453, 527, 609, 1037, 1041, 1067, 1074, 1182, 1220, 1246, 1254, 1298, 1364		
<code>\@esphack</code>	412, 432, 1479		
<code>\@for</code>	419		
<code>\@ifclassloaded</code>	1143		
<code>\@ifdefinable</code>	262		
<code>\@ifnextchar</code>	384, 978		
<code>\@ifpackageloaded</code> ..	1163, 1171, 1179		
<code>\@ifstar</code>	373, 1210		
<code>\@ifundefined</code>	192, 579, 616, 643, 661, 680, 718, 749, 837, 852, 961, 988, 1034, 1199, 1296, 1318, 1434, 1445, 1493		
<code>\@input</code>	1338		
<code>\@inputcheck</code>	1231, 1232, 1244, 1262, 1264, 1287		
<code>\@latex@warning</code>	499		
<code>\@mainaux</code>	949		
<code>\@namedef</code>	387		
<code>\@ne</code>	1001, 1512		
<code>\@newl@bel</code>	258		
<code>\@nil</code>	879, 881, 1067, 1074, 1256, 1257, 1348, 1358		
<code>\@part</code>	1108		
<code>\@schapter</code>	1132		
<code>\@sect</code>	1120		
<code>\@spart</code>	1126		
<code>\@ssect</code>	1138		
<code>\@testopt</code>	1212, 1215, 1223		
<code>\@tfor</code>	293, 308, 454		
<code>\@undefined</code>	1026, 1347		
<code>\@</code>	27, 28, 29, 30, 153, 155, 157, 158, 170, 173, 1322, 1350, 1376, 1385, 1392, 1401		
A			
<code>\AddLineBeginAux</code>	254		
<code>\advance</code>	685, 1001, 1512		
<code>\afterassignment</code>	757, 761		
<code>\AfterLastShipout</code>	682		
<code>\Alph</code>	7		
<code>\anchor</code>	1390		
<code>\AtBeginDocument</code>	664, 843, 943, 1100, 1575		
<code>\AtBeginShipout</code>	650, 723		
<code>\AtEndOfPackage</code>	195		
B			
<code>\beamer@section</code>	1145		
<code>\beamer@subsection</code>	1151		
<code>\beamer@subsubsection</code>	1157		
<code>\begin</code>	25, 59, 102, 108, 156, 172, 1547, 1571, 1576, 1581, 1582		
<code>\bfseries</code>	598		
C			
<code>\c@abspage</code>	653, 1007, 1012		
<code>\c@page</code>	685, 963, 1005		
<code>\c@zpage</code>	1005, 1009		
<code>\c@zref@unique</code> ..	585, 1001, 1512, 1514		
<code>\chapter</code>	26, 32, 34, 63, 84		
<code>\ChapterPages</code>	93, 113		
<code>\ChapterStart</code>	80, 135, 150, 166		
<code>\ChapterStop</code>	87, 148, 165, 184		
<code>\chardef</code>	779, 794, 803, 807		
<code>\cleaders</code>	1534		
<code>\cleardoublepage</code>	81, 88		
<code>\clearpage</code>	64		
<code>\closein</code>	1244		
<code>\comma@entry</code> ..	885, 886, 887, 893, 896		
<code>\comma@parse</code>	884		
<code>\csname</code>	223, 226, 227, 263, 273, 287, 312, 326, 340, 359, 388, 389, 393, 396, 399, 409, 456, 458, 463, 464, 480, 507, 518, 519, 521, 540, 541, 542, 566, 914, 922, 973, 989, 991, 994, 996, 1007, 1011, 1015, 1016, 1018, 1020, 1021, 1026, 1076, 1079, 1208, 1307, 1320, 1329, 1414, 1457		
<code>\current@chapid</code>	82, 90		
D			
<code>\DeclareOption</code>	194		
<code>\default</code>	1369, 1404		
<code>\define@key</code>	1075, 1078, 1081, 1084, 1422, 1497, 1500, 1503		
<code>\detokenize</code>	1062		
<code>\dftest</code>	167, 174, 175, 176, 177, 178, 179, 180, 181, 182		
<code>\dimen@</code>	1517, 1522, 1534		
<code>\dimexpr</code>	153, 155, 1522		
<code>\do</code>	294, 313, 419, 454		
<code>\documentclass</code> ..	4, 41, 70, 246, 1540, 1565		
<code>\dotfill</code>	169, 173		
E			
<code>\emph</code>	150		
<code>\end</code>	36, 66, 130, 159, 183, 185, 1560, 1573, 1578, 1584, 1585		
<code>\endcsname</code>	223, 226, 227, 263, 273, 287, 312, 326, 340, 359, 388, 389, 393, 396, 399, 409, 456, 458, 463, 464, 480, 507, 518, 519, 521, 540, 541, 542, 553, 566, 914, 922, 972, 973, 989, 991, 994, 995, 996, 1007, 1011, 1015, 1016, 1018, 1020, 1021, 1026, 1076, 1079, 1208, 1307, 1320, 1329, 1414, 1457		
<code>\endinput</code>	192, 238, 251, 616, 643, 661, 680, 718, 749, 837, 852, 961, 1034, 1199, 1434, 1445, 1451, 1464, 1493		
<code>\etex@unexpanded</code>	354, 547, 1283		
<code>\Expect</code>	1548, 1551, 1554, 1556, 1558, 1559, 1572, 1577, 1583		
<code>\ExpectRuns</code>	1577, 1583		

F		<code>\lst@label</code> 1182
<code>\fancyhead</code> 53, 56		<code>\lst@MakeCaption</code> 1181
<code>\fancyhf</code> 52, 55		<code>\LT@c@ption</code> 1173
<code>\fancypagestyle</code> 54		<code>\ltx@empty</code> 867
<code>\filename@area</code> 1284		<code>\ltx@firstofone</code> . . . 228, 549, 551, 554
<code>\filename@parse</code> 1222		<code>\ltx@firstoftwo</code>
<code>\foo</code> 20, 31, 33, 35		. 274, 302, 360, 481, 530, 562, 699
<code>\frontmatter</code> 60, 105		<code>\ltx@gobble</code> . . . 224, 618, 619, 898, 1096
G		<code>\ltx@gobbletwo</code> 581, 646, 900
<code>\g@addto@macro</code>		<code>\ltx@ifUndefined</code> . 231, 589, 728, 1446
. . . 326, 348, 994, 1328, 1369,		<code>\ltx@ifundefined</code> 1203, 1436
1372, 1381, 1390, 1397, 1403, 1414		<code>\ltx@secondoftwo</code> 276, 304,
<code>\G@refundefinedtrue</code> 498		362, 483, 515, 528, 556, 560, 701
<code>\gdef</code> 393, 950, 989, 991		<code>\ltx@space</code> . . . 800, 806, 859, 887, 930
<code>\GetTitleStringDisableCommands</code> 1071		<code>\ltx@zero</code> 857, 910
<code>\GetTitleStringExpand</code> 1057	M	
<code>\GetTitleStringNonExpand</code> 1059	<code>\m@ne</code> 685	
<code>\GetTitleStringResult</code> 1062	<code>\mainmatter</code> 62, 134	
H		<code>\makeatletter</code> . 11, 76, 103, 1209, 1546
<code>\hb@xt@</code> 1534	<code>\makeatother</code> 18, 101	
<code>\hfill</code> 1523, 1534	<code>\makebox</code> 169, 170	
<code>\hss</code> 1534	<code>\MessageBreak</code>	
I		. 244, 860, 863, 890, 892, 931,
<code>\if@filesw</code> 444, 683, 948, 1475		932, 1234, 1300, 1341, 1448, 1461
<code>\if@safe@actives</code> 559	N	
<code>\ifcase</code> 116, 815, 885	<code>\NeedsTeXFormat</code> 3, 188,	
<code>\ifcsname</code> 553, 972, 995	214, 612, 639, 657, 674, 714,	
<code>\ifeof</code> 1232, 1264, 1287	745, 833, 848, 939, 957, 1030,	
<code>\ifetex@unexpanded</code> 241, 345	1195, 1430, 1441, 1489, 1538, 1564	
<code>\ifnum</code> 697,	<code>\newcommand</code> 20, 80, 87,	
790, 800, 806, 857, 910, 1459, 1521	93, 167, 486, 617, 624, 633, 636,	
<code>\ifodd</code> 125	704, 739, 756, 770, 773, 774,	
<code>\ifpdf</code> 1454	840, 854, 977, 1025, 1087, 1090,	
<code>\ifx</code> 223, 273, 296, 359, 455, 480, 526,	1206, 1425, 1481, 1484, 1509, 1510	
618, 867, 876, 880, 885, 886,	<code>\newcounter</code> 6, 582, 647, 968	
887, 1015, 1182, 1254, 1273,	<code>\newif</code> 220, 434, 1040, 1053, 1205	
1277, 1281, 1298, 1322, 1347,	<code>\newlabel</code> 1314, 1324, 1337	
1350, 1376, 1385, 1392, 1401, 1457	<code>\newmarks</code> 863	
<code>\ifZREF@found</code> . . . 220, 301, 1367, 1374	<code>\newpage</code> 143	
<code>\ifZREF@immediate</code> . 434, 446, 450, 456	<code>\nfss@text</code> 598	
<code>\ifzref@titleref@expand</code> . 1040, 1056	<code>\nofiles</code> 1539	
<code>\ifzref@titleref@stripperiod</code> . . .	<code>\number</code> 96,	
. 1053, 1064	111, 585, 590, 729, 944, 963,	
<code>\ifZREF@xr@zreflabel</code> 1205, 1239, 1272	983, 1007, 1012, 1514, 1577, 1583	
<code>\immediate</code> 439, 949	<code>\numexpr</code> 96,	
<code>\IncludeTests</code> 1543, 1568	111, 116, 592, 731, 787, 944,	
<code>\item</code> 109, 112, 114, 122, 126, 128	983, 1294, 1303, 1316, 1344, 1521	
K		O
<code>\kern</code> 1531	<code>\on@line</code> 946, 1246	
<code>\kv@key</code> 879, 881, 882, 896	<code>\openin</code> 1231	
<code>\kv@parse</code> 875	P	
<code>\kv@value</code> 876, 877, 884	<code>\PackageError</code> 232,	
L		243, 267, 281, 367, 858, 1447, 1460
<code>\l</code> 417	<code>\PackageInfo</code> . 264, 382, 947, 1238, 1247	
<code>\label</code> 618, 1096	<code>\PackageWarning</code>	
<code>\leavevmode</code> 1511 321, 336, 421, 888, 929, 1233	
<code>\LogTests</code> 1544, 1569	<code>\PackageWarningNoLine</code> . . . 1299, 1340	
<code>\lst@caption</code> 1185	<code>\page</code> 1372	
	<code>\pagestyle</code> 51	

\pdflastxpos	1469	\url	1397, 1416
\pdflastypos	1470	\usepackage	9, 43, 50, 72, 74, 1541, 1542, 1566, 1567
\pdfsavepos	1448, 1461, 1476, 1515, 1528		
\pdfTeXversion	1459		
\ProcessOptions	211	V	
\protect	498	\value	13, 725
\protected@write	449	\verb	173
\providecommand			
	255, 942, 1202, 1506, 1507, 1508	W	
\ProvidesPackage		\write	438, 439, 949
	189, 215, 613, 640, 658, 675, 715, 746, 834, 849, 940, 958, 1031, 1196, 1431, 1442, 1490	X	
R		\x	293, 295, 296, 419, 420, 422, 426, 567, 569, 813, 829, 912, 917, 920, 926, 982, 985, 1270, 1273, 1277, 1281, 1295, 1297, 1305, 1307, 1317, 1321
\read	1262	\XR@ext	1203
\refstepcounter	666		
\renewcommand	7, 48, 584	Y	
\RequirePackage	191, 196, 217, 240, 245, 253, 615, 642, 644, 660, 677, 678, 679, 717, 719, 720, 748, 750, 751, 752, 753, 836, 838, 839, 851, 853, 960, 962, 1033, 1035, 1036, 1198, 1200, 1433, 1444, 1453, 1492, 1494, 1496	\y	292, 296
\reset@font	598	Z	
\rightarrow	47	\z@	978, 1531
\romannumeral	868	\Z@D@page	774
		\Z@L@alist	1551, 1556, 1559
S		\Z@L@LastPage	688
\section	65, 137, 145	\Z@L@main	687, 1548, 1554, 1558
\setcounter	649	\zdotfill	15, 170, 173, 1510
\setkeys	1088, 1426, 1509	\zdotfillsetup	15, 1509
\setlength	1517	\zexternaldocument	16, 1206
\space	500, 1249, 1250, 1448, 1461	\ziflastpage	10, 704
\stepcounter	21, 651, 970, 971	\zifrefundefined	7, 486
		\zlabel	9, 85, 106, 138, 146, 617
T		\zmakeperpage	13, 977
\tableofcontents	61, 132	\znexpage	12, 53, 56, 770
\the	13, 153, 155, 426, 431, 462, 476, 592, 653, 691, 725, 731, 787, 1050, 1284, 1294, 1303, 1316, 1344, 1469, 1470	\znexpagesetup	12, 44, 756
\thechapter	14	\znonexpagename	48, 773, 821
\thefoo	7, 12, 22	\zpageref	9, 127, 633
\thetype	1329	\zposx	15, 153, 1481, 1521
\thepage	45, 46, 47, 447, 451, 500, 606, 1006	\zposy	15, 155, 1481
\thezpage	14, 1006, 1010	\zref	9, 27, 28, 29, 30, 113, 115, 124, 129, 139, 624, 634
\thezref@unique		\ZREF@newprop	389, 392
	9, 584, 1003, 1004, 1009, 1010, 1012, 1516, 1518, 1521, 1529	\ZREF@makeperpage	978, 983, 987
\title	1381, 1405	\ZREF@newprop	384, 386
\toks@	418, 425, 426, 431, 460, 462, 475, 476, 686, 691, 1044, 1050, 1271, 1284	\ZREF@perpage@step	992, 1000
\TR@TitleReference	1366, 1400	\zref@addprop	
\ttl@sect@i	1165		5, 15, 16, 17, 78, 317, 607, 608, 654, 663, 722, 896, 965, 966, 967, 1039, 1438, 1471, 1472
		\ZREF@addtoks	474
U		\ZREF@baseok	609
\unexpanded	244, 249	\ZREF@call	779, 794, 803, 807, 815
\UniqueCounterCall	771	\zref@default	7, 384, 595, 597
\UniqueCounterNew	754	\ZREF@df@dot	1504, 1508, 1534
		\ZREF@df@min	1501, 1506, 1522
		\ZREF@df@unit	1498, 1507, 1517
		\ZREF@dotfill	1519, 1525, 1533
		\ZREF@extract	505, 510
		\zref@extract	6, 97, 98, 111, 140, 504, 631, 734, 825, 1009, 1010, 1097, 1482, 1485

<code>\zref@extractdefault</code>	<code>\ZREF@np@call@next</code>	765, 769, 824
..... 6, 117, 118, 511,	<code>\ZREF@np@call@nonext</code> ..	762, 768, 820
534, 697, 698, 783, 798, 841, 1012	<code>\ZREF@np@call@unknown</code> .	758, 767, 816
<code>\ZREF@foundfalse</code>	<code>\ZREF@np@setup@i</code>	757, 760
291, 1365	<code>\ZREF@np@setup@ii</code>	761, 764
<code>\ZREF@foundtrue</code>	<code>\ZREF@number</code>	589, 856
297, 1407	<code>\ZREF@org@@caption</code>	1104
<code>\zref@getcurrent</code>	<code>\ZREF@org@@chapter</code>	1116
6, 398	<code>\ZREF@org@@part</code>	1110
<code>\ZREF@gttemp</code>	<code>\ZREF@org@@schapter</code>	1134
347, 348, 349	<code>\ZREF@org@@sect</code>	1122
<code>\ZREF@iflastpage</code>	<code>\ZREF@org@@spart</code>	1128
705, 707, 707	<code>\ZREF@org@@ssect</code>	1140
<code>\zref@iflastpage</code>	<code>\ZREF@org@beamer@section</code>	1147
10, 696, 710	<code>\ZREF@org@beamer@subsection</code> ...	1153
<code>\ZREF@iflistcontainsprop</code> ...	<code>\ZREF@org@beamer@subsubsection</code>	1159
287, 289	<code>\ZREF@org@lst@MakeCaption</code>	1188
<code>\zref@iflistcontainsprop</code>	<code>\ZREF@org@LT@c@ption</code>	1174
..... 5, 286, 320, 335	<code>\ZREF@org@refstepcounter</code>	668
<code>\zref@iflistundefined</code>	<code>\ZREF@org@stepcounter</code>	970, 975
5, 261, 272, 280	<code>\ZREF@org@thepage</code>	447, 451
<code>\zref@ifpropundefined</code>	<code>\ZREF@org@ttl@sect@i</code>	1167
..... 6, 358, 366, 420,	<code>\ZREF@org@write</code>	438, 439
909, 1325, 1359, 1378, 1387, 1394	<code>\ZREF@P</code>	383, 387, 388, 389,
<code>\ZREF@ifrefcontainsprop</code> ...	390, 393, 454, 456, 458, 463, 464	
517, 525	<code>\ZREF@pagenum@last</code>	797, 800
<code>\zref@ifrefcontainsprop</code>	<code>\ZREF@pagenum@this</code>	
..... 7, 513, 1411, 1412 782, 787, 790, 800, 806	
<code>\ZREF@ifrefundefined</code>	<code>\ZREF@patch</code> ..	221, 665, 1101, 1107,
..... 487, 489, 780, 791, 801	1113, 1119, 1125, 1131, 1137,	
<code>\zref@ifrefundefined</code>	1144, 1150, 1156, 1164, 1172, 1180	
7, 479,	<code>\zref@prop</code>	309, 314
491, 497, 514, 535, 792, 1004, 1518	<code>\zref@propexists</code> 6, 319, 334, 365, 625	
<code>\ZREF@immediatetrue</code>	<code>\ZREF@refname@next</code> .	785, 792, 801, 825
437	<code>\ZREF@refname@this</code> .	777, 778, 780, 783
<code>\ZREF@l@addto@macro</code>	<code>\ZREF@refused</code>	494, 496
340, 345	<code>\zref@refused</code>	7, 490, 493,
<code>\ZREF@label</code>	630, 636, 708, 709, 737, 844, 1095	
407, 431, 443, 691	<code>\zref@require@unique</code> 8, 578, 969, 1495	
<code>\zref@label</code>	<code>\zref@setcurrent</code> .	5, 83, 390, 395, 667
6, 401, 621	<code>\zref@setdefault</code>	7, 594, 597
<code>\zref@labelbylist</code>	<code>\zref@setmainlist</code>	7, 600
..... 6, 402, 404, 725, 1003, 1477	<code>\ZREF@STAR</code>	880, 904
<code>\zref@labelbyprops</code>	<code>\ZREF@stripperiod</code>	1066, 1074
..... 6, 90, 414, 778, 1516, 1529	<code>\ZREF@temp</code>	193, 200,
<code>\zref@listexists</code> 5, 279, 318, 333, 406	201, 202, 203, 204, 205, 206,	
<code>\zref@listforloop</code>	207, 208, 209, 210, 325, 327,	
307	453, 460, 461, 469, 879, 880, 1514	
<code>\zref@localaddprop</code> .	<code>\ZREF@TempName</code>	855, 867,
5, 332, 1553, 1555	868, 870, 896, 909, 913, 921, 932	
<code>\ZREF@mainlist</code>	<code>\ZREF@TempNum</code>	
402, 601, 856, 857, 861, 868, 910, 923	
604, 607, 608, 654, 663, 1039, 1438	<code>\zref@thepage</code>	11, 733, 741
<code>\ZREF@makeperpage@opt</code>	<code>\zref@thepage@name</code>	
978, 980 11, 728, 734, 737, 786	
<code>\ZREF@MARKS@DefineProp</code>	<code>\zref@thepage@refused</code>	736, 740
..... 872, 873, 874, 908	<code>\ZREF@titleref</code>	1091, 1093
<code>\zref@marks@register</code> 854, 859, 891, 930	<code>\zref@titleref@cleanup</code> ..	1042, 1082
<code>\ZREF@name</code>	<code>\zref@titleref@current</code>	
218,	1037, 1038, 1061, 1065, 1066, 1085	
232, 243, 267, 281, 321, 336,	<code>\ZREF@titleref@hook</code>	
367, 421, 858, 888, 929, 1447, 1460 1041, 1045, 1049, 1072	
<code>\ZREF@NAME@bot</code>		
887, 907		
<code>\ZREF@NAME@first</code>		
886, 906		
<code>\ZREF@NAME@top</code>		
885, 905		
<code>\zref@newlabel</code>		
..... 6, 255, 257, 469, 1292, 1336		
<code>\zref@newlist</code>		
5, 260,		
604, 681, 721, 882, 964, 1468, 1550		
<code>\ZREF@newprop</code>		
375, 378, 381		
<code>\zref@newprop</code>		
..... 5, 12, 13, 14, 77, 372, 605,		
606, 653, 662, 913, 921, 963,		
1038, 1201, 1326, 1360, 1379,		
1388, 1395, 1435, 1469, 1470, 1549		
<code>\ZREF@nextpage</code>		
771, 775		
<code>\ZREF@nil</code> 393, 527, 542, 1263, 1269,		
1274, 1278, 1292, 1305, 1314,		
1321, 1346, 1353, 1363, 1366, 1400		
<code>\ZREF@NOVALUE</code>		
533		
<code>\ZREF@novalue</code>		
526, 527, 533		

<code>\zref@titleref@setcurrent</code>	<code>\ZREF@xr@line</code> ..
..... 1055 , 1103 , 1109 ,	1262 , 1263 , 1274 , 1278
1115 , 1121 , 1127 , 1133 , 1139 ,	<code>\ZREF@xr@list</code>
1146 , 1152 , 1158 , 1166 , 1175 , 1184	1297 , 1298
<code>\zref@titleref@stripperiodtrue</code> 1054	<code>\ZREF@xr@newlabel</code>
<code>\ZREF@unexpanded</code> 536 , 538 , 547 , 549 , 551	1277 , 1337
<code>\ZREF@UpdatePdfTeX</code> ... 219 , 1450 , 1463	<code>\ZREF@xr@prefix</code>
<code>\ZREF@wrapper@babel</code>	1219 ,
569 , 575	1293 , 1309 , 1311 , 1315 , 1331 , 1333
<code>\zref@wrapper@babel</code>	<code>\ZREF@xr@procesfile</code>
8 , 140 ,	1266
487 , 494 , 552 , 621 , 626 , 705 , 1091	<code>\ZREF@xr@process@label</code> ..
<code>\zref@wrapper@immediate</code>	1278 , 1314
..... 8 , 89 , 435 , 690 , 724	<code>\ZREF@xr@process@zreflabel</code> 1274 , 1292
<code>\zref@wrapper@unexpanded</code>	<code>\ZREF@xr@processfile</code> 1230 , 1261 , 1289
8 , 546	<code>\ZREF@xr@processline</code>
<code>\ZREF@X</code>	1263 , 1269
374 , 377 , 388	<code>\ZREF@xr@refname</code>
<code>\ZREF@xr@input</code>	1293 ,
1281 , 1338	1296 , 1307 , 1315 , 1318 , 1320 , 1329
<code>\ZREF@xr@checkfile</code> ..	<code>\ZREF@xr@scanparams</code>
1227 , 1230 , 1258	1319 , 1363
<code>\ZREF@xr@checkkey</code>	<code>\ZREF@xr@scantitleref</code> ...
1348 , 1358	1366 , 1400
<code>\ZREF@xr@checklist</code>	<code>\ZREF@xr@url</code>
1305 , 1346	1226 , 1416
<code>\zref@xr@ext</code>	<code>\ZREF@xr@urlcheck</code> ..
16 , 1202 , 1223 , 1423	1309 , 1331 , 1410
<code>\ZREF@xr@externaldocument</code>	<code>\ZREF@xr@zref@newlabel</code> ..
..... 1212 , 1215 , 1218	1273 , 1336
<code>\ZREF@xr@file</code>	<code>\ZREF@xr@zreflabelfalse</code>
1221 , 1231 ,	1211
1234 , 1239 , 1248 , 1256 , 1301 , 1342	<code>\ZREF@xr@zreflabeltrue</code>
<code>\ZREF@xr@filelist</code>	1214
1220 , 1254 , 1256 , 1257 , 1282 , 1283	<code>\ZREF@zref</code>
<code>\ZREF@xr@found</code> .	626 , 629
1241 , 1249 , 1294 , 1316	<code>\zrefused</code> .
<code>\ZREF@xr@graburl</code>	9 , 94 , 95 , 161 , 162 , 163 , 636
1223 , 1225	<code>\zruns</code>
<code>\ZREF@xr@ignored</code> 1242 , 1250 , 1303 , 1344	13 , 942 , 1572 , 1577 , 1583
<code>\ZREF@xr@ignorewarning</code>	<code>\zsavepos</code>
..... 1311 , 1333 , 1339	15 , 157 , 158 , 1473
	<code>\zthepage</code>
	11 , 739
	<code>\ztitleref</code>
	14 , 1090
	<code>\ztitlerefsetup</code>
	14 , 1075
	<code>\ztotpages</code>
	13 , 125 , 840
	<code>\zunknownnextpagename</code> ..
	12 , 774 , 817
	<code>\zunmakeperpage</code>
	14 , 1025
	<code>\zxrsetup</code>
	16 , 1422