The zref package

Heiko Oberdiek
<heiko.oberdiek at googlemail.com>

2011/03/18 v2.21

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

Standard LaTeX's reference system with `\label`, `\ref`, and `\pageref` supports two properties, the apperance of the counter that is last incremented by `\refstepcounter` and the page with the `\label` command.

Unhappily LaTeX 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 LaTeX behaviour

References are created by the `\label` command:

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

Now LaTeX records the section number `2.1` and the page `7` in the reference. Internally the reference is a list with two entries:

```
\r@myref → {2.1}{7}
```

The length of the list if fixed in the LaTeX kernel, An interface for adding new properties is missing.

There are several tries to add new properties:

hyperref uses a list of five properties instead of the standard list with two entries. This causes many compatibility problems with LaTeX and other packages.

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

1.2 Basic idea

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

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

The entries have the following format:

`\xref_`⟨*your key*⟩`_key{`⟨*some text*⟩`}`

This approach is much more flexible:

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

Unhappily I am not familiar with the experimental code for LaTeX3 that will need some time before its first release. Thus I have implemented it as LaTeX 2_ε package without disturbing the existing LaTeX reference system.

1.3 Interfaces

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

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

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

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

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

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

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

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

2 Interface for programmers

The user interface is described in the next section 3.

2.1 Entities

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

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

The generic format of a entry is:

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

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

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

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

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

2.2 Property list

$^{\text{exp}}$ means that the implementation of the marked macro is expandable. $^{\text{exp2}}$ goes a step further and marks the macro expandable in exact two expansion steps.

`\zref@newlist {`⟨*listname*⟩`}`

Declares a new empty property list.

`\zref@addprop {`⟨*listname*⟩`} {`⟨*propname list*⟩`}`

Adds the properties of ⟨*propname list*⟩ (comma separated) to the property list ⟨*listname*⟩. The property and list must exist. A ⟨*propname list*⟩ can be given since 2010/04/19 v2.13. Before this version only one property name could be added in one call of `\zref@addprop`.

`\zref@localaddprop {`⟨*listname*⟩`} {`⟨*propname list*⟩`}`

Local variant of `\zref@addprop`.

`\zref@listexists` {⟨*listname*⟩} {⟨*then*⟩}

Executes ⟨*then*⟩ if the property list ⟨*listname*⟩ exists or raise an error otherwise.

`\zref@iflistundefined`$^{\text{exp}}$ {⟨*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 LaTeX's `\currentlabel`.

`\zref@getcurrent`$^{\text{exp2}}$ {⟨*propname*⟩}

This returns the current value of the property ⟨*propname*⟩. The value may not be correct, especially if the property is bound to a page (start form of `\zref@newprop`) and the right value is only known at shipout time (e.g. property 'page'). In case of errors (e.g. unknown property) the empty string is returned.

Since version 2010/04/22 v2.14 `\zref@getcurrent` supports `\zref@wrapper@unexpanded`.

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

Calls ⟨*then*⟩ if the property ⟨*propname*⟩ is available or generates an error message otherwise.

`\zref@ifpropundefined`$^{\text{exp}}$ {⟨*propname*⟩} {⟨*then*⟩} {⟨*else*⟩}

Calls ⟨*then*⟩ or ⟨*else*⟩ depending on the existence of property ⟨*propname*⟩.

2.4 Reference generation

`\zref@label` {⟨*refname*⟩}

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

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

Same as `\zref@label` except that the properties are taken from the specified property list ⟨*listname*⟩.

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

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

`\zref@newlabel` {⟨*refname*⟩} `{`...`}`

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

2.5 Data extraction

`\zref@extractdefault`$^{\text{exp2}}$ {⟨*refname*⟩} {⟨*propname*⟩} {⟨*default*⟩}

This is the basic command that refernces the value of a property ⟨*propname*⟩ for the reference ⟨*refname*⟩. In case of errors such as undefined reference the ⟨*default*⟩ is used instead.

`\zref@extract`$^{\text{exp2}}$ {⟨*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@ifrefundefined`$^{\text{exp}}$ {⟨*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@ifrefcontainsprop`$^{\text{exp}}$ {⟨*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
(base)	`default`	`main`	<*empty*>
	`page`	`main`	<*empty*>
abspage	`abspage`	`main`	0
counter	`counter`	`main`	<*empty*>
hyperref	`anchor`	`main`	<*empty*>
	`url`		<*empty*>
pageattr	`pdfpageattr`	`thepage`	`...`
	`pdfpagesattr`	`LastPage`	`...`
pagelayout[1]	`mag`	`thepage`	`\number\mag`
	`paperwidth`	`thepage`	`\number\paperwidth`
	`paperheight`	`thepage`	`\number\paperheight`
	`stockwidth`	`thepage`	`\number\stockwidth`
	`stockheight`	`thepage`	`\number\stockheight`
	`pdfpageheight`	`thepage`	`\number\pdfpageheight`
	`pdfpagewidth`	`thepage`	`\number\pdfpagewidth`
	`pdfhorigin`	`thepage`	`\number\pdfhorigin`
	`pdfvorigin`	`thepage`	`\number\pdfvorigin`
	`hoffset`	`thepage`	`\number\hoffset`
	`voffset`	`thepage`	`\number\voffset`
	`topmargin`	`thepage`	`\number\topmargin`
	`oddsidemargin`	`thepage`	`\number\oddsidemargin`
	`evensidemargin`	`thepage`	`\number\evensidemargin`
	`textwidth`	`thepage`	`\number\textwidth`
	`textheight`	`thepage`	`\number\textheight`
	`headheight`	`thepage`	`\number\headheight`
	`headsep`	`thepage`	`\number\headsep`
	`footskip`	`thepage`	`\number\footskip`
	`marginparwidth`	`thepage`	`\number\marginparwidth`
	`marginparsep`	`thepage`	`\number\marginparsep`
	`columnwidth`	`thepage`	`\number\columnwidth`
	`columnsep`	`thepage`	`\number\columnsep`
perpage	`pagevalue`	`perpage`	0
	`page`	`perpage`	<*empty*>
	`abspage`	`perpage`	0
savepos	`posx`	`savepos`	0
	`posy`	`savepos`	0
titleref	`title`	`main`	<*empty*>
xr	`anchor`		<*empty*>
	`externaldocument`		<*empty*>
	`theotype`		<*empty*>
	`title`		<*empty*>
	`url`		<*empty*>

2.8 Wrapper for advanced situations

`\zref@wrapper@babel {...} {⟨name⟩}`

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

[1]Module pagelayout only defines properties if the parameter exists.

`\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:

```
\zref@wrapper@unexpanded{%
  \edef\foo{%
    \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. Supported macros are `\zref@extract`, `\zref@extractdefault` and since version 2010/04/22 v2.14 macro `\zref@getcurrent`.

2.9 Counter for unique names

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

`\zref@require@unique`

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

`\thezref@unique`

This command is used to generate unique label names.

3 User interface

3.1 Module user

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

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

babel	Babel shorthands are allowed.
robust	Robust macro.
exp	Expandable version: • robust, unless the extracted values are fragile, • no babel shorthand suport.
exp2	Expandable like $^{\text{exp}}$ and: • expandable in exact two steps.

The basic user interface of the package without modules are commands that mimic the standard LaTeX behaviour of `\label`, `\ref`, and `\pageref`:

`\zlabel {⟨refname⟩}`$^{\text{babel}}$

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

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

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

`\zref{x}` ≡ `\zref[default]{x}`

`\zpageref {⟨refname⟩}`$^{\text{babel}}$

Convenience macro, similar to `\pageref`.

`\zpageref{x}` ≡ `\zref[page]{x}`

`\zrefused {⟨refname⟩}`$^{\text{babel}}$

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@iflastpage`$^{\text{exp}}$ {⟨*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

```
⟨*example-lastpage⟩
%<<END_EXAMPLE
\NeedsTeXFormat{LaTeX2e}
\documentclass{report}

\newcounter{foo}
\renewcommand*{\thefoo}{\Alph{foo}}

\usepackage{zref-lastpage,zref-user}[2011/03/18]

\makeatletter
\zref@newprop{thefoo}{\thefoo}
\zref@newprop{valuefoo}{\the\value{foo}}
\zref@newprop{chapter}{\thechapter}
\zref@addprops{LastPage}{thefoo,valuefoo,chapter}
\makeatother

\newcommand*{\foo}{%
  \stepcounter{foo}%
  [Current foo: \thefoo]%
}

\begin{document}
  \chapter{First chapter}
  Last page is \zref{LastPage}.\\
  Last chapter is \zref[chapter]{LastPage}.\\
  Last foo is \zref[thefoo]{LastPage}.\\
  Last value of foo is \zref[valuefoo]{LastPage}.\\
  \foo
  \chapter{Second chapter}
  \foo\foo\foo
  \chapter{Last chapter}
  \foo
\end{document}
%END_EXAMPLE
```

36 ⟨/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@name`$^{\text{exp}}$ {⟨*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@thepage`$^{\text{exp}}$ {⟨*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⟩}`

Acording 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

```
⟨*example-nextpage⟩
%<<END_EXAMPLE
\documentclass{book}

\usepackage{zref-nextpage}[2011/03/18]
\znextpagesetup
  {\thepage}% next page is unknown
  {\thepage\ (#1)}% this page is last page
  {\thepage\ $\rightarrow$ #1}% next page is known
\renewcommand*{\znonextpagename}{last page}

\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhf{}
\fancyhead[LE,RO]{\znextpage}
\fancypagestyle{plain}{%
  \fancyhf{}%
  \fancyhead[LE,RO]{\znextpage}%
}

\begin{document}
\frontmatter
  \tableofcontents
\mainmatter
  \chapter{Hello World}
  \clearpage
  \section{Last section}
\end{document}
%END_EXAMPLE
⟨/example-nextpage⟩
```

3.6 Module **totpages**

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

`\ztotpages`$^{\text{exp}}$

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

3.7 Module pagelayout

The module defines additional properties for each parameter of the page layout that is effective during page shipout. The value of length parameters is given in sp without the unit as plain number.

Some parameters are specific for a class (e.g. stockwidth and stockheight for class memoir) or the TeX engine like pdfTeX. If the parameter is not available, then the property will not be defined. The default value of the property is the current setting of the parameter.

The module thepage is loaded that generates a label for each page. The properties of module pagelayout are added to the property list `thepage` of module thepage.

List of properties:

parameter	property	remarks
`\mag`	mag	
`\paperwidth`	paperwidth	
`\paperheight`	paperheight	
`\stockwidth`	stockwidth	class memoir
`\stockheight`	stockheight	class memoir
`\pdfpagewidth`	pdfpagewidth	pdfTeX, LuaTeX
`\pdfpageheight`	pdfpageheight	pdfTeX, LuaTeX
`\pdfhorigin`	pdfhorigin	pdfTeX, LuaTeX
`\pdfvorigin`	pdfvorigin	pdfTeX, LuaTeX
`\hoffset`	hoffset	
`\voffset`	voffset	
`\topmargin`	topmargin	
`\oddsidemargin`	oddsidemargin	
`\evensidemargin`	evensidemargin	
`\textwidth`	textwidth	
`\textheight`	textheight	
`\headheight`	headheight	
`\headsep`	headsep	
`\footskip`	footskip	
`\marginparwidth`	marginparwidth	
`\marginparsep`	marginparsep	
`\columnwidth`	columnwidth	
`\columnsep`	columnsep	

```
\zlistpagelayout
```

At the end of document the page layout parameter for each page are printed into the `.log` file if macro `\zlistpagelayout` is called before `\end{document}` (preamble is a good place).

3.8 Module marks

ToDo.

3.9 Module runs

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

`\zruns`$^{\text{exp}}$

Prints the 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`).

3.10 Module perpage

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

3.11 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.

3.12 Module titleref

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

`\ztitleref {⟨refname⟩}`$^{\text{babel}}$

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.

`stripperiod=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.

3.13 Module savepos

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

`\zsavepos {⟨refname⟩}`

It generates a reference with name ⟨*refname*⟩. The reference stores the location where `\zsavepos` is executed in properties `posx` and `posy`.

`\zposx`$^{\text{exp}}$ `{⟨refname⟩}`
`\zposy`$^{\text{exp}}$ `{⟨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 LATEX 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.

`\zref@savepos`

Macro `\zref@savepos` performs the first part of `\zsavepos` by calling `\pdfsavepos` (if `.aux` files are writable).

Thus `\zsavepos` is basically `\zref@savepos` followed by `\zreflabelbylist{`⟨*refname*⟩`}{savepos}`. The property list `savepos` contains the properties `posx` and `posy`.

3.14 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 {`key_1=$value_1$, key_2=$value_2$, ...`}`

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

`min=`⟨*count value*⟩
: If the actual number of dots are smaller than ⟨*count value*⟩, then the dots are suppressed. Default: `2`.

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

`dot=`⟨*value*⟩
: The dot itself is given by ⟨*value*⟩. Default: `.` (dot, same as the dot in `\dotfill`).

3.15 Module **env**

This module defines two properties `envname` and `envline`. They remember the name of the environment and the line number at the start of the environment.

3.16 Module **xr**

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

`\zexternaldocument * [`⟨*prefix*⟩`]`$^{\text{babel}}$ `{`⟨*external document*⟩`} [`⟨*url*⟩`]`

See `\externaldocument` for a description of this option. The found labels also get a property `externaldocument` that remembers ⟨*external document*⟩. The standard reference scheme and the scheme of this package use different name spaces for reference names. If the external document uses both systems. Then one import statement would put the names in one namespace and probably causing problems with multiple references of the same name. Thus the star form only looks for `\newlabel` in the `.aux` files, whereas without star only `\zref@newlabel`s 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 {`*key*$_1$=*value*$_1$, *key*$_2$=*value*$_2$, ...`}`

The following setup options are available:

ext: It sets the default extension.

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

toltxlabel: Boolean option. The found references are imported as LaTeX labels. Packages nameref, titleref and class memoir are supported.

urluse: Boolean option. If enabled, then a URL is stored in a macro and the macro is put in property 'urluse'. The URL is not put in property 'url'. The purpose is to save TeX memory.

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

`\zref@xr@ext`

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

4 ToDo

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

- Other applications: autoref, hyperref, ...

5 Example

```
⟨*example⟩
\documentclass{book}

\usepackage[ngerman]{babel}%

\usepackage[savepos,totpages,titleref,dotfill,counter,user]{zref}

```

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.

```
\makeatletter
\zref@newprop{chaptitle}{}
\zref@addprop{main}{chaptitle}

\newcommand*{\ChapterStart}[2]{%
  \cleardoublepage
  \def\current@chapid{#1}%
  \zref@setcurrent{chaptitle}{#2}%
  \chapter{#2}%
  \zlabel{chap:#1}%
}
```

```
\newcommand*{\ChapterStop}{%
  \cleardoublepage
  \zref@wrapper@immediate{%
    \zref@labelbyprops{chapend:\current@chapid}{abspage}%
  }%
}
```

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

```
\newcommand*{\ChapterPages}[1]{%
  \zrefused{chap:#1}%
  \zrefused{chapend:#1}%
  \number\numexpr
    \zref@extract{chapend:#1}{abspage}%
    -\zref@extract{chap:#1}{abspage}%
    +1\relax
}
\makeatother
```

```
\begin{document}
```

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

```
\makeatletter

\frontmatter
\zlabel{documentstart}

\begin{itemize}
\item
  The frontmatter part has
  \number\numexpr\zref@extract{chap:first}{abspage}-1\relax
  ~pages.
\item
  Chapter \zref{chap:first} has \ChapterPages{first} page(s).
\item
  Section \zref{hello} is on the
  \ifcase\numexpr
    \zref@extractdefault{hello}{page}{0}%
    -\zref@extractdefault{chap:first}{page}{0}%
    +1\relax
    ??\or first\or second\or third\or forth\fi
  ~page inside its chapter.
\item
  The document has
  \zref[abspage]{LastPage} pages.
  This number is \ifodd\ztotpages odd\else even\fi.
\item
  The last page is labeled with \zpageref{LastPage}.
\item
  The title of chapter \zref{chap:next} %
  is ``\zref[chaptitle]{chap:next}''.
\end{itemize}

\tableofcontents

\mainmatter
\ChapterStart{first}{First chapter}

```

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.

```
\section{Test}
\zlabel{a"o}
```

```
Section \zref{a"o} on page
\zref@wrapper@babel\zref@extract{a"o}{page}.

Text.
\newpage

\section{Hello World}
\zlabel{hello}

\ChapterStop

\ChapterStart{next}{Next chapter with \emph{umlauts}: "a"o"u"s}

```

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 LaTeX compile run.

```
The width of the first column is
  \the\dimexpr \zposx{secondcol}sp - \zposx{firstcol}sp\relax,\\
the height difference of the two baselines is
  \the\dimexpr \zposy{firstcol}sp - \zposy{secondline}sp\relax:\\
\begin{tabular}{ll}
  \zsavepos{firstcol}Hello&\zsavepos{secondcol}World\\
  \zsavepos{secondline}Second line&foobar\\
\end{tabular}

```

With `\zrefused` LaTeX is notified, if the references are not yet available and LaTeX can generate the rerun hint.

```
\zrefused{firstcol}
\zrefused{secondcol}
\zrefused{secondline}

\ChapterStop
```

Test for module `\dotfill`.

```
\ChapterStart{dotfill}{Test for dotfill feature}
\newcommand*{\dftest}[1]{%
  #1&
  [\makebox[{#1}]{\dotfill}]&
  [\makebox[{#1}]{\zdotfill}]\\
}
\begin{tabular}{rll}
& [\verb|\dotfill|] & [\verb|\zdotfill|]\\
\dftest{0.43em}
\dftest{0.44em}
\dftest{0.45em}
\dftest{0.87em}
\dftest{0.88em}
\dftest{0.89em}
\dftest{1.31em}
\dftest{1.32em}
\dftest{1.33em}
\end{tabular}
\ChapterStop
\end{document}
```

⟨/example⟩

6 Implementation

6.1 Package **zref**

6.1.1 Identification

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref}
  [2011/03/18 v2.21 New reference scheme for LaTeX2e (HO)]%
```

6.1.2 Load basic module

```
\RequirePackage{zref-base}[2011/03/18]
```

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

```
\@ifundefined{ZREF@base@ok}{\endinput}{}
```

6.1.3 Process options

Known modules are loaded and the release date is checked.

```
\def\ZREF@temp#1{%
  \DeclareOption{#1}{%
    \AtEndOfPackage{%
      \RequirePackage{zref-#1}[2011/03/18]%
    }%
  }%
}
\ZREF@temp{abspage}
\ZREF@temp{counter}
\ZREF@temp{dotfill}
\ZREF@temp{hyperref}
\ZREF@temp{lastpage}
\ZREF@temp{marks}
\ZREF@temp{nextpage}
\ZREF@temp{pageattr}
\ZREF@temp{pagelayout}
\ZREF@temp{perpage}
\ZREF@temp{runs}
\ZREF@temp{savepos}
\ZREF@temp{thepage}
\ZREF@temp{titleref}
\ZREF@temp{totpages}
\ZREF@temp{user}
\ZREF@temp{xr}

\ProcessOptions\relax
⟨/package⟩
```

6.2 Module base

6.2.1 Prefixes

This package uses the following prefixes for macro names:

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

`\ZREF@`: Internal macros.

`\Z@L@`*`listname`*: The properties of the list ⟨*listname*⟩.

`\Z@D@`*`propname`*: The default value for property ⟨*propname*⟩.

`\Z@E@`*`propname`*: Extract function for property ⟨*propname*⟩.

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

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

`\Z@R@`*`labelname`*: Data for reference ⟨*labelname*⟩.

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

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

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

ZREF@TR: Setup for module titleref.

6.2.2 Identification

```
⟨*base⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-base}%
  [2011/03/18 v2.21 Module base for zref (HO)]%
```

6.2.3 Utilities

```
\RequirePackage{ltxcmds}[2010/03/01]
\RequirePackage{infwarerr}[2010/04/08]
\RequirePackage{kvsetkeys}[2010/03/01]
\RequirePackage{kvdefinekeys}[2010/03/01]
\RequirePackage{pdftexcmds}[2010/04/01]
```

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

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

```
\ltx@IfUndefined{protected}{%
  \RequirePackage{makerobust}[2006/03/18]%
```

\ZREF@Robust

```
  \def\ZREF@Robust#1#2{%
    \def\ZREF@temp{\MakeRobustcommand#2}%
    \afterassignment\ZREF@temp
    #1#2%
  }%
```

```
}{%
```

\ZREF@Robust

```
  \def\ZREF@Robust#1{%
    \protected#1%
  }%
```

```
}
```

\ZREF@IfDefinable

```
\def\ZREF@IfDefinable#1#2#3{%
  \@ifdefinable{#1}{%
    \ZREF@Robust{#2}#1#3%
  }%
}
```

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

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

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

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

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

```
\def\ZREF@patch#1{%
  \ltx@IfUndefined{#1}{%
    \ltx@gobble
  }{%
    \expandafter\let\csname ZREF@org@#1\expandafter\endcsname
    \csname #1\endcsname
    \ltx@firstofone
  }%
}
```

6.2.4 Check for ε-TeX

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

```
\ltx@IfUndefined{eTeXversion}{%
  \PackageError\ZREF@name{%
    Missing support for eTeX; package is abandoned%
  }{%
    Use a TeX compiler that support eTeX and enable eTeX %
    in the format.%
  }%
  \endinput
}{}%
```

```
\RequirePackage{etexcmds}[2007/09/09]
\ifetex@unexpanded
\else
  \PackageError\ZREF@name{%
    Missing e-TeX's \string\unexpanded.\MessageBreak
    Add \string\RequirePackage\string{etexcmds\string} before %
    \string\documentclass%
  }{%
    Probably you are using some package (e.g. ConTeXt) that %
    redefines \string\unexpanded%
  }%
  \expandafter\endinput
\fi
```

6.2.5 Auxiliary file stuff

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

```
\RequirePackage{auxhook}
\AddLineBeginAux{%
  \string\providecommand\string\zref@newlabel[2]{}%
}
```

`\ZREF@RefPrefix`

```
\def\ZREF@RefPrefix{Z@R}
```

`\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.
- LaTeX's check for multiple references.
- LaTeX's check for changed references.

```
\ZREF@Robust\edef\zref@newlabel{%
  \noexpand\@newl@bel{\ZREF@RefPrefix}%
}
```

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.

```
\ZREF@Robust\def\zref@newlist#1{%
  \zref@iflistundefined{#1}{%
    \@ifdefinable{Z@L@#1}{%
      \global\expandafter\let\csname Z@L@#1\endcsname\ltx@empty
      \PackageInfo\ZREF@name{New property list: #1}%
    }%
  }{%
    \PackageError\ZREF@name{%
      Property list `#1' already exists%
    }\@ehc
  }%
}
```

\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.

```
\def\zref@iflistundefined#1{%
  \ltx@ifundefined{Z@L@#1}%
}
```

\zref@listexists

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

```
\ZREF@Robust\def\zref@listexists#1{%
  \zref@iflistundefined{#1}{%
    \PackageError\ZREF@name{%
      Property list `#1' does not exist%
    }\@ehc
  }%
}
```

\zref@iflistcontainsprop

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

```
\ZREF@Robust\def\zref@iflistcontainsprop#1#2{%
  \zref@iflistundefined{#1}{%
    \ltx@secondoftwo
  }{%
    \begingroup\expandafter\endgroup
    \expandafter\in@
    \csname#2\expandafter\expandafter\expandafter\endcsname
    \expandafter\expandafter\expandafter{\csname Z@L@#1\endcsname}%
    \csname ltx@\ifin@ first\else second\fi oftwo\endcsname
  }%
}
```

\zref@listforloop

```
\def\zref@listforloop#1#2{%
  \zref@listexists{#1}{%
    \expandafter\expandafter\expandafter\@tfor
    \expandafter\expandafter\expandafter\zref@prop
    \expandafter\expandafter\expandafter:%
    \expandafter\expandafter\expandafter=%
    \csname Z@L@#1\endcsname
    \do{%
      \begingroup
        \escapechar=-1 %
        \edef\x{\endgroup
          \def\noexpand\zref@prop{%
            \expandafter\string\zref@prop
          }%
        }%
      \x
      #2\zref@prop
    }%
  }%
}
```

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

```
\ZREF@Robust\def\zref@addprops#1#2{%
  \zref@listexists{#1}{%
    \comma@parse{#2}{%
      \zref@propexists\comma@entry{%
        \zref@iflistcontainsprop{#1}\comma@entry{%
          \PackageWarning\ZREF@name{%
            Property `\comma@entry' is already in list `#1'%
          }%
        }{%
          \begingroup\expandafter\endgroup
          \expandafter\g@addto@macro
          \csname Z@L@#1\expandafter\endcsname
          \expandafter{\csname\comma@entry\endcsname}%
        }%
      }%
      \ltx@gobble
    }%
  }%
}
```

\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.

```
\ZREF@Robust\def\zref@addprop#1#2{%
  \zref@listexists{#1}{%
    \zref@propexists{#2}{%
      \zref@iflistcontainsprop{#1}{#2}{%
        \PackageWarning\ZREF@name{%
          Property `#2' is already in list `#1'%
        }%
      }{%
        \begingroup\expandafter\endgroup
        \expandafter\g@addto@macro
        \csname Z@L@#1\expandafter\endcsname
        \expandafter{\csname#2\endcsname}%
      }%
    }%
  }%
}
```

\zref@localaddprops

```
\ZREF@Robust\def\zref@localaddprops#1#2{%
  \zref@listexists{#1}{%
    \comma@parse{#2}{%
      \zref@propexists\comma@entry{%
        \zref@iflistcontainsprop{#1}\comma@entry{%
          \PackageWarning\ZREF@name{%
            Property `\comma@entry' is already in list `#1'%
          }%
        }{%
          \begingroup\expandafter\endgroup
          \expandafter\ltx@LocalAppendToMacro
          \csname Z@L@#1\expandafter\endcsname
          \expandafter{\csname\comma@entry\endcsname}%
        }%
      }%
      \ltx@gobble
    }%
  }%
}
```

\zref@localaddprop

```
\ZREF@Robust\def\zref@localaddprop#1#2{%
  \zref@listexists{#1}{%
    \zref@propexists{#2}{%
      \zref@iflistcontainsprop{#1}{#2}{%
        \PackageWarning\ZREF@name{%
          Property `#2' is already in list `#1'%
        }%
      }{%
        \begingroup\expandafter\endgroup
        \expandafter\ltx@LocalAppendToMacro
        \csname Z@L@#1\expandafter\endcsname
        \expandafter{\csname#2\endcsname}%
      }%
    }%
  }%
}
```

```
\ltx@IfUndefined{pdf@strcmp}{%
```

\zref@delprop

```
  \ZREF@Robust\def\zref@delprop{%
    \ZREF@delprop\gdef
  }%
```

\zref@localdelprop

```
  \ZREF@Robust\def\zref@localdelprop{%
    \ZREF@delprop\def
  }%
```

\ZREF@delprop

```
  \def\ZREF@delprop#1#2#3{%
    \zref@listexists{#2}{%
      \begingroup
        \escapechar=-1 %
        \def\ZREF@param{#3}%
        \@onelevel@sanitize\ZREF@param
        \toks@{}%
        \expandafter\expandafter\expandafter\ZREF@@delprop
        \csname Z@L@#2\endcsname!%
      \expandafter\endgroup
      \expandafter#1\csname Z@L@#2\expandafter\endcsname
      \expandafter{%
        \the\toks@
      }%
    }%
  }%
```

\ZREF@@delprop

```
  \def\ZREF@@delprop#1{%
    \expandafter\ZREF@@@delprop\expandafter{\string#1}#1%
  }%
```

\ZREF@@@delprop

```
  \def\ZREF@@@delprop#1#2{%
    \ifx#2!%
    \else
      \def\ZREF@temp{#1}%
      \@onelevel@sanitize\ZREF@temp
      \ifx\ZREF@param\ZREF@temp
      \else
```

```
        \toks@\expandafter{%
          \the\expandafter\toks@\csname#1\endcsname
        }%
      \fi
      \expandafter\ZREF@@delprop
    \fi
  }%
}{%
```

\zref@delprop

```
  \ZREF@Robust\def\zref@delprop{%
    \ZREF@delprop\xdef
  }%
```

\zref@localdelprop

```
  \ZREF@Robust\def\zref@localdelprop{%
    \ZREF@delprop\edef
  }%
```

\ZREF@delprop

```
  \def\ZREF@delprop#1#2#3{%
    \zref@listexists{#2}{%
      \def\ZREF@param{#3}%
      \edef\ZREF@SavedEscapechar{\the\escapechar}%
      \escapechar=-1 %
      \expandafter#1\csname Z@L@#2%
      \expandafter\expandafter\expandafter\endcsname{%
        \expandafter\expandafter\expandafter\ZREF@@delprop
        \csname Z@L@#2\endcsname!%
      }%
      \escapechar=\ZREF@SavedEscapechar\relax
    }%
  }%
```

\ZREF@@delprop Caution: #1 might be an \if or similar token.

```
  \def\ZREF@@delprop#1{%
    \expandafter\ZREF@@@delprop\expandafter{\string#1}#1%
  }%
```

\ZREF@@@delprop

```
  \def\ZREF@@@delprop#1#2{%
    \ifx#2!%
    \else
      \ifnum\pdf@strcmp{#1}{\ZREF@param}=\ltx@zero
      \else
        \expandafter\noexpand\csname#1\endcsname
      \fi
      \expandafter\ZREF@@delprop
    \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.

```
\def\zref@ifpropundefined#1{%
  \ltx@ifundefined{Z@E@#1}%
}
```

`\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.

```
\ZREF@Robust\def\zref@propexists#1{%
  \zref@ifpropundefined{#1}{%
    \PackageError\ZREF@name{%
      Property `#1' does not exist%
    }\@ehc
  }%
}
```

`\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.

```
\ZREF@Robust\def\zref@newprop{%
  \@ifstar{%
    \let\ZREF@X\noexpand
    \ZREF@newprop
  }{%
    \let\ZREF@X\ltx@empty
    \ZREF@newprop
  }%
}
```

`\ZREF@newprop`

```
\def\ZREF@newprop#1{%
  \edef\ZREF@P{#1}%
  \@onelevel@sanitize\ZREF@P
  \begingroup
  \ifx\ZREF@P\ZREF@par
    \@PackageError\ZREF@name{%
      Invalid property name `\ZREF@P'%
    }{%
      The property name `par' is not allowed %
      because of internal reasons.%
      \MessageBreak
      \@ehc
    }%
    \def\ZREF@@newprop[##1]##2{\endgroup}%
  \else
    \zref@ifpropundefined\ZREF@P{%
      \endgroup
      \PackageInfo\ZREF@name{%
        New property: \ZREF@P
      }%
    }{%
      \@PackageError\ZREF@name{%
        Property `\ZREF@P' already exists%
      }\@ehc
      \def\ZREF@@newprop[##1]##2{\endgroup}%
    }%
  \fi
  \@ifnextchar[\ZREF@@newprop{\ZREF@@newprop[\zref@default]}%
}
```

\ZREF@par

```
\def\ZREF@par{par}
\@onelevel@sanitize\ZREF@par
```

\ZREF@@newprop

```
\def\ZREF@@newprop[#1]{%
  \global\@namedef{Z@D@\ZREF@P}{#1}%
  \global\expandafter\let\csname Z@X@\ZREF@P\endcsname\ZREF@X
  \begingroup\expandafter\endgroup
  \expandafter\ZREF@@@newprop\csname\ZREF@P\endcsname
  \expandafter\gdef\csname Z@C@\ZREF@P\endcsname{}%
  \zref@setcurrent\ZREF@P
}
\def\ZREF@@@newprop#1{%
  \expandafter
  \gdef\csname Z@E@\ZREF@P\endcsname##1#1##2##3\ZREF@nil{##2}%
}
```

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

```
\ZREF@Robust\def\zref@setcurrent#1#2{%
  \zref@propexists{#1}{%
    \expandafter\def\csname Z@C@#1\endcsname{#2}%
  }%
}
```

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

```
\def\ZREF@getcurrent#1{%
  \romannumeral0%
  \ltx@ifundefined{Z@C@#1}{%
    \ltx@space
  }{%
    \expandafter\expandafter\expandafter\ltx@space
    \csname Z@C@#1\endcsname
  }%
}
```

\ZREF@u@getcurrent

```
\def\ZREF@wu@getcurrent#1{%
  \etex@unexpanded\expandafter\expandafter\expandafter{%
    \ZREF@getcurrent{#1}%
  }%
}
```

\zref@getcurrent

```
\let\zref@getcurrent\ZREF@getcurrent
```

6.2.8 Reference generation

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

```
\ZREF@Robust\def\zref@label#1{%
  \zref@labelbylist{#1}\ZREF@mainlist
}
```

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

```
\ZREF@Robust\def\zref@labelbylist#1#2{%
  \@bsphack
    \zref@listexists{#2}{%
      \expandafter\expandafter\expandafter\ZREF@label
      \expandafter\expandafter\expandafter{%
        \csname Z@L@#2\endcsname
    }{#1}%
  }%
  \@esphack
}
```

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

```
\ZREF@Robust\def\zref@labelbyprops#1#2{%
  \@bsphack
    \begingroup
      \toks@{}%
      \comma@parse{#2}{%
        \zref@ifpropundefined\comma@entry{%
          \PackageWarning\ZREF@name{%
            Property `\comma@entry' is not known%
          }%
        }{%
          \toks@\expandafter{%
            \the\expandafter\toks@\csname\comma@entry\endcsname
          }%
        }%
        \ltx@gobble
      }%
    \expandafter\endgroup
    \expandafter\ZREF@label\expandafter{\the\toks@}{#1}%
  \@esphack
}
```

`\zref@labelbykv`

```
\ZREF@Robust\def\zref@labelbykv#1#2{%
  \@bsphack
    \begingroup
      \let\Z@L@ZREF@temp\ltx@empty
      \kvsetkeys{ZREF@LABEL}{#1}%
      \ifZREF@immediate
        \expandafter\zref@wrapper@immediate\expandafter{%
          \expandafter\ZREF@label\expandafter{\Z@L@ZREF@temp}{#2}%
        }%
      \else
        \expandafter\ZREF@label\expandafter{\Z@L@ZREF@temp}{#2}%
      \fi
    \endgroup
  \@esphack
}
```

```
\kv@define@key{ZREF@LABEL}{prop}{%
  \edef\ZREF@param{#1}%
  \zref@propexists\ZREF@param{%
    \zref@iflistcontainsprop{ZREF@temp}\ZREF@param{}{%
      \begingroup\expandafter\endgroup
      \expandafter\ltx@LocalAppendToMacro
      \expandafter\Z@L@ZREF@temp
      \expandafter{\csname\ZREF@param\endcsname}%
    }%
  }%
}
\kv@define@key{ZREF@LABEL}{list}{%
  \zref@listforloop{#1}{%
    \zref@iflistcontainsprop{ZREF@temp}\zref@prop{}{%
      \begingroup\expandafter\endgroup
      \expandafter\ltx@LocalAppendToMacro
      \expandafter\Z@L@ZREF@temp
      \expandafter{\csname\zref@prop\endcsname}%
    }%
    \ltx@gobble
  }%
}
\kv@define@key{ZREF@LABEL}{delprop}{%
  \zref@propexists{#1}{%
    \zref@localdelprop{ZREF@temp}{#1}%
  }%
}
\kv@define@key{ZREF@LABEL}{immediate}[true]{%
  \edef\ZREF@param{#1}%
  \ifx\ZREF@param\ZREF@true
    \ZREF@immediatetrue
  \else
    \ifx\ZREF@param\ZREF@false
      \ZREF@immediatefalse
    \else
      \PackageWarning\ZREF@name{%
        Option `immediate' expects `true' or `false'.\MessageBreak
        Ignoring invalid value `\ZREF@param'%
      }%
    \fi
  \fi
}
```

`\ZREF@false`

```
\def\ZREF@false{false}
```

`\ZREF@true`

```
\def\ZREF@true{true}
```

```
\kv@define@key{ZREF@LABEL}{values}[]{%
  \kv@parse{#1}{%
    \ifx\kv@value\relax
      \@PackageWarning\ZREF@name{%
        Missing value for property `\kv@key'%
      }%
      \expandafter\ltx@gobbletwo
    \else
      \expandafter\zref@setcurrent
    \fi
  }%
}
```

`\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.

```
\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`.

```
\ZREF@Robust{\long\def}\zref@wrapper@immediate#1{%
  \begingroup
    \ZREF@immediatetrue
    \let\ZREF@org@write\write
    \def\write{\immediate\ZREF@org@write}%
    #1%
  \endgroup
}
```

\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 32is made expandable in this case.

```
\def\ZREF@label#1#2{%
  \if@filesw
    \begingroup
      \ifZREF@immediate
        \let\ZREF@org@thepage\thepage
      \fi
      \protected@write\@auxout{%
        \ifZREF@immediate
          \let\thepage\ZREF@org@thepage
        \fi
        \let\ZREF@temp\ltx@empty
        \@tfor\ZREF@P:=#1\do{%
          \begingroup
            \escapechar=-1 %
            \edef\x{\endgroup
              \def\noexpand\ZREF@P{%
                \expandafter\string\ZREF@P
              }%
            }%
          \x
          \expandafter\ifx
              \csname
                \ifZREF@immediate
                  relax%
                \else
                  Z@X@\ZREF@P%
                \fi
              \endcsname
              \noexpand
            \expandafter\let\csname Z@C@\ZREF@P\endcsname\relax
          \fi
          \toks@\expandafter{\ZREF@temp}%
          \edef\ZREF@temp{%
            \the\toks@
            \ltx@backslashchar\ZREF@P{%
              \expandafter\noexpand\csname Z@C@\ZREF@P\endcsname
            }%
          }%
        }%
      }{%
        \string\zref@newlabel{#2}{\ZREF@temp}%
      }%
    \endgroup
  \fi
}
\def\ZREF@addtoks#1{%
  \toks@\expandafter\expandafter\expandafter{%
    \expandafter\the\expandafter\toks@#1%
  }%
}
```

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 LaTeX 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 pdfTeX 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.

```
\def\zref@ifrefundefined#1{%
  \ltx@ifundefined{Z@R@#1}%
}
```

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

```
\ZREF@IfDefinable\zifrefundefined\def{%
  #1{%
    \zref@wrapper@babel\ZREF@ifrefundefined{#1}%
  }%
}
```

`\ZREF@ifrefundefined`

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

`\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 LaTeX and a warning is given.

```
\ZREF@Robust\def\zref@refused#1{%
  \zref@wrapper@babel\ZREF@refused{#1}%
}
```

`\ZREF@refused`

```
\def\ZREF@refused#1{%
  \zref@ifrefundefined{#1}{%
    \protect\G@refundefinedtrue
    \@latex@warning{%
      Reference `#1' on page \thepage \space undefined%
    }%
  }{}%
}
```

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

```
\def\zref@ifrefcontainsprop#1#2{%
  \zref@ifrefundefined{#1}{%
    \ltx@secondoftwo
  }{%
    \expandafter\ZREF@ifrefcontainsprop
    \csname Z@E@#2\expandafter\endcsname
    \csname#2\expandafter\expandafter\expandafter\endcsname
    \expandafter\expandafter\expandafter{%
      \csname Z@R@#1\endcsname
    }%
  }%
}
\def\ZREF@ifrefcontainsprop#1#2#3{%
  \expandafter\ifx\expandafter\ZREF@novalue
  #1#3#2\ZREF@novalue\ZREF@nil\ltx@empty
    \expandafter\ltx@secondoftwo
  \else
    \expandafter\ltx@firstoftwo
  \fi
}
\def\ZREF@novalue{\ZREF@NOVALUE}
```

\zref@extract

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

```
\def\ZREF@extract#1#2{%
  \romannumeral0%
  \ltx@ifundefined{Z@D@#2}{%
    \expandafter\ltx@space\zref@default
  }{%
    \expandafter\expandafter\expandafter\ZREF@@extract
    \expandafter\expandafter\expandafter{%
      \csname Z@D@#2\endcsname
    }{#1}{#2}%
  }%
}
```

\ZREF@@extract

```
\def\ZREF@@extract#1#2#3{%
  \expandafter\expandafter\expandafter\ltx@space
  \zref@extractdefault{#2}{#3}{#1}%
}
```

\ZREF@wu@extract

```
\def\ZREF@wu@extract#1#2{%
  \etex@unexpanded\expandafter\expandafter\expandafter{%
    \ZREF@extract{#1}{#2}%
  }%
}
```

\zref@extract

```
\let\zref@extract\ZREF@extract
```

\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.

```
\def\ZREF@extractdefault#1#2#3{%
  \romannumeral0%
  \zref@ifrefundefined{#1}\ltx@firstoftwo{%
    \zref@ifpropundefined{#2}\ltx@firstoftwo\ltx@secondoftwo
  }{%
    \ltx@space
    #3%
  }{%
    \expandafter\expandafter\expandafter\ltx@space
    \csname Z@E@#2\expandafter\expandafter\expandafter\endcsname
    \csname Z@R@#1\expandafter\endcsname
    \csname#2\endcsname{#3}\ZREF@nil
  }%
}
```

\ZREF@wu@extractdefault

```
\def\ZREF@wu@extractdefault#1#2#3{%
  \etex@unexpanded\expandafter\expandafter\expandafter{%
    \ZREF@extractdefault{#1}{#2}{#3}%
  }%
}
```

\zref@extractdefault

```
\let\zref@extractdefault\ZREF@extractdefault
```

\ZREF@wrapper@unexpanded

```
\ZREF@Robust{\long\def}\ZREF@wrapper@unexpanded#1{%
  \let\zref@wrapper@unexpanded\ltx@firstofone
  \let\zref@getcurrent\ZREF@wu@getcurrent
  \let\zref@extractdefault\ZREF@wu@extractdefault
  \let\zref@extract\ZREF@wu@extract
  #1%
  \let\zref@wrapper@unexpanded\ZREF@wrapper@unexpanded
  \let\zref@getcurrent\ZREF@getcurrent
  \let\zref@extractdefault\ZREF@extractdefault
  \let\zref@extract\ZREF@extract
}
```

\zref@wrapper@unexpanded

```
\ltx@IfUndefined{etex@unexpanded}{%
  \let\zref@wrapper@unexpanded\ltx@firstofone
}{%
  \let\zref@wrapper@unexpanded\ZREF@wrapper@unexpanded
}
```

6.2.10 Compatibility with babel

\zref@wrapper@babel

```
\ZREF@Robust{\long\def}\zref@wrapper@babel#1#2{%
  \ifcsname if@safe@actives\endcsname
    \expandafter\ltx@firstofone
  \else
    \expandafter\ltx@secondoftwo
  \fi
  {%
    \if@safe@actives
      \expandafter\ltx@secondoftwo
    \else
      \expandafter\ltx@firstoftwo
    \fi
    {%
      \begingroup
        \csname @safe@activestrue\endcsname
        \edef\x{#2}%
      \expandafter\endgroup
      \expandafter\ZREF@wrapper@babel\expandafter{\x}{#1}%
    }%
  }{%
    #1{#2}%
  }%
}
\long\def\ZREF@wrapper@babel#1#2{%
  #2{#1}%
}
```

6.2.11 Unique counter support

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

```
\ZREF@Robust\def\zref@require@unique{%
  \@ifundefined{c@zref@unique}{%
    \begingroup
      \let\@addtoreset\ltx@gobbletwo
      \newcounter{zref@unique}%
    \endgroup
```

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

```
    \renewcommand*{\thezref@unique}{%
      zref@\number\c@zref@unique
    }%
  }{}%
}
```

6.2.12 Utilities

\ZREF@number

```
\ltx@IfUndefined{numexpr}{%
  \let\ZREF@number\number
}{%
  \def\ZREF@number#1{\the\numexpr#1}%
}
```

6.2.13 Setup

\zref@setdefault Standard LaTeX 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`.

```
\ZREF@Robust\def\zref@setdefault#1{%
  \def\zref@default{#1}%
}
```

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

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

Main property list.

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

```
\ZREF@Robust\def\zref@setmainlist#1{%
  \def\ZREF@mainlist{#1}%
}
\zref@setmainlist{main}
```

Now we create the list.

```
\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 LaTeX 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

```
\zref@newprop{default}{\@currentlabel}
\zref@newprop*{page}{\thepage}
\zref@addprops\ZREF@mainlist{default,page}
```

Properties

\ZREF@NewPropAnchor

```
\def\ZREF@NewPropAnchor{%
  \zref@newprop{anchor}{%
    \ltx@ifundefined{@currentHref}{}{\@currentHref}%
  }%
  \global\let\ZREF@NewPropAnchor\relax
}
```

\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`.

\ZREF@NewPropTitle

```
\def\ZREF@NewPropTitle{%
  \gdef\zref@titleref@current{}%
  \zref@newprop{title}{\zref@titleref@current}%
  \global\let\ZREF@NewPropTitle\relax
}
```

\ZREF@NewPropTheotype

```
\def\ZREF@NewPropTheotype{%
  \zref@newprop{theotype}{}%
  \global\let\ZREF@NewPropTheotype\relax
}
```

Mark successful loading

```
\let\ZREF@base@ok=Y
⟨/base⟩
```

6.3 Module user

```
⟨*user⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-user}%
  [2011/03/18 v2.21 Module user for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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

First we define the pendants to the standard LaTeX 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.

```
\newcommand*\zlabel{%
  \ifx\label\ltx@gobble
    \expandafter\ltx@gobble
  \else
    \expandafter\zref@wrapper@babel\expandafter\zref@label
  \fi
}%
```

\zkvlabel

```
\newcommand*{\zkvlabel}[1]{%
  \ifx\label\ltx@gobble
    \expandafter\ltx@gobblethree
  \fi
  \zref@wrapper@babel{\zref@labelbykv{#1}}%
}%
```

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

```
\newcommand*{\zref}[2][default]{% robust because of optional argument
  \zref@propexists{#1}{%
    \zref@wrapper@babel\ZREF@zref{#2}{#1}%
  }%
}%
\def\ZREF@zref#1{%
  \zref@refused{#1}%
  \zref@extract{#1}%
}%
```

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

```
\ZREF@IfDefinable\zpageref\def{%
  {\zref[page]}%
}
```

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

```
\ZREF@IfDefinable\zrefused\def{%
  {\zref@refused}%
}
```

```
⟨/user⟩
```

6.4 Module **abspage**

```
⟨*abspage⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-abspage}%
  [2011/03/18 v2.21 Module abspage for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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.

```
\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 TeX files.

```
\begingroup
  \let\@addtoreset\ltx@gobbletwo
  \newcounter{abspage}%
\endgroup
\setcounter{abspage}{0}%
\AtBeginShipout{%
  \stepcounter{abspage}%
}%
\zref@newprop*{abspage}[0]{\the\c@abspage}%
\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.

```
⟨/abspage⟩
```

6.5 Module counter

```
⟨*counter⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-counter}%
  [2011/03/18 v2.21 Module counter for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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.

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

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

```
\AtBeginDocument{%
  \ZREF@patch{refstepcounter}{%
    \def\refstepcounter#1{%
      \zref@setcurrent{counter}{#1}%
      \ZREF@org@refstepcounter{#1}%
    }%
  }%
}
⟨/counter⟩
```

6.6 Module lastpage

```
⟨*lastpage⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-lastpage}%
  [2011/03/18 v2.21 Module lastpage for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\RequirePackage{zref-abspage}[2011/03/18]
\RequirePackage{atveryend}[2009/12/07]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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.

```
\zref@newlist{LastPage}
\AfterLastShipout{%
  \if@filesw
    \begingroup
      \advance\c@page\m@ne
      \toks@\expandafter\expandafter\expandafter{%
        \expandafter\Z@L@main
        \Z@L@LastPage
      }%
      \expandafter\zref@wrapper@immediate\expandafter{%
        \expandafter\ZREF@label\expandafter{\the\toks@}{LastPage}%
      }%
    \endgroup
  \fi
}
```

`\zref@iflastpage`

```
\def\zref@iflastpage#1{%
```

```
  \ifnum\zref@extractdefault{#1}{abspage}{-1}=%
        \zref@extractdefault{LastPage}{abspage}{-2} %
    \expandafter\ltx@firstoftwo
  \else
    \expandafter\ltx@secondoftwo
  \fi
}
```

\ziflastpage

```
\ZREF@IfDefinable\ziflastpage\def{%
  {\zref@wrapper@babel\ZREF@iflastpage}%
}
```

ZREF@iflastpage

```
\def\ZREF@iflastpage#1{%
  \zref@refused{LastPage}%
  \zref@refused{#1}%
  \zref@iflastpage{#1}%
}
```

⟨/lastpage⟩

6.7 Module **thepage**

⟨*thepage⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-thepage}%
  [2011/03/18 v2.21 Module thepage for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi

\RequirePackage{atbegshi}
\RequirePackage{zref-abspage}[2011/03/18]

\zref@newlist{thepage}
\zref@addprop{thepage}{page}
\AtBeginShipout{%
  \zref@wrapper@immediate{%
    \zref@labelbylist{thepage\the\value{abspage}}{thepage}%
  }%
}
```

\zref@thepage@name

```
\ltx@IfUndefined{numexpr}{%
  \def\zref@thepage@name#1{thepage\number#1}%
}{%
  \def\zref@thepage@name#1{thepage\the\numexpr#1}%
}
```

\zref@thepage

```
\def\zref@thepage#1{%
  \zref@extract{\zref@thepage@name{#1}}{page}%
}%
```

\zref@thepage@refused

```
\ZREF@Robust\def\zref@thepage@refused#1{%
  \zref@refused{\zref@thepage@name{#1}}%
}%
```

\zthepage

```
\ZREF@IfDefinable\zthepage\def{%
  #1{%
    \zref@thepage@refused{#1}%
    \zref@thepage{#1}%
  }%
}
```

```
⟨/thepage⟩
```

6.8 Module nextpage

```
⟨*nextpage⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-nextpage}%
  [2011/03/18 v2.21 Module nextpage for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\RequirePackage{zref-abspage}[2011/03/18]
\RequirePackage{zref-thepage}[2011/03/18]
\RequirePackage{zref-lastpage}[2011/03/18]
\RequirePackage{uniquecounter}[2009/12/18]
```

```
\UniqueCounterNew{znextpage}

\newcommand*{\znextpagesetup}{%
  \afterassignment\ZREF@np@setup@i
  \def\ZREF@np@call@unknown##1%
}
\def\ZREF@np@setup@i{%
  \afterassignment\ZREF@np@setup@ii
  \def\ZREF@np@call@nonext##1%
}
\def\ZREF@np@setup@ii{%
  \def\ZREF@np@call@next##1%
}
\def\ZREF@np@call@unknown#1{#1}
\def\ZREF@np@call@nonext#1{#1}
\def\ZREF@np@call@next#1{#1}
\ZREF@IfDefinable\znextpage\def{%
  {\UniqueCounterCall{znextpage}{\ZREF@nextpage}}%
}%
\newcommand*{\znonextpagename}{}
\newcommand*{\zunknownnextpagename}{\Z@D@page}
\def\ZREF@nextpage#1{%
  \begingroup
    \def\ZREF@refname@this{zref@np#1}%
    \zref@labelbyprops\ZREF@refname@this{abspage}%
    \chardef\ZREF@call=0 % unknown
    \ZREF@ifrefundefined\ZREF@refname@this{%
    }{%
      \edef\ZREF@pagenum@this{%
        \zref@extractdefault\ZREF@refname@this{abspage}{0}%
      }%
      \edef\ZREF@refname@next{%
        \zref@thepage@name{%
          \the\numexpr\ZREF@pagenum@this+1%
        }%
      }%
      \ifnum\ZREF@pagenum@this>0 %
```

```
        \ZREF@ifrefundefined{LastPage}{%
          \zref@ifrefundefined\ZREF@refname@next{%
          }{%
            \chardef\ZREF@call=2 % next page
          }%
        }{%
          \edef\ZREF@pagenum@last{%
            \zref@extractdefault{LastPage}{abspage}{0}%
          }%
          \ifnum\ZREF@pagenum@this<\ZREF@pagenum@last\ltx@space
            \ZREF@ifrefundefined\ZREF@refname@next{%
            }{%
              \chardef\ZREF@call=2 % next page
            }%
          \else
            \ifnum\ZREF@pagenum@this=\ZREF@pagenum@this\ltx@space
              \chardef\ZREF@call=1 % no next page
            \fi
          \fi
        }%
      \fi
    }%
    \edef\x{%
      \endgroup
      \ifcase\ZREF@call
        \noexpand\ZREF@np@call@unknown{%
          \noexpand\zunknownnextpagename
        }%
      \or
        \noexpand\ZREF@np@call@nonext{%
          \noexpand\znonextpagename
        }%
      \else
        \noexpand\ZREF@np@call@next{%
          \noexpand\zref@extract{\ZREF@refname@next}{page}%
        }%
      \fi
    }%
  \x
}
⟨/nextpage⟩
```

6.9 Module totpages

```
⟨*totpages⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-totpages}%
  [2011/03/18 v2.21 Module totpages for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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

```
\RequirePackage{zref-abspage}[2011/03/18]
\RequirePackage{zref-lastpage}[2011/03/18]
```

\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.

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

Also we mark the reference LastPage as used:

```
\AtBeginDocument{%
  \zref@refused{LastPage}%
}
```

```
⟨/totpages⟩
```

6.10 Module **pagelayout**

```
⟨*pagelayout⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-pagelayout}%
  [2011/03/18 v2.21 Module pagelayout for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\RequirePackage{zref-thepage}[2011/03/18]
\RequirePackage{ifluatex}[2010/03/01]
\RequirePackage{atveryend}[2010/03/24]
```

```
\ifluatex
  \ifnum\luatexversion<39 %
  \else
    \begingroup
      \escapechar=-1 %
      \def\ZREF@temp#1{%
        \ltx@IfUndefined{\string#1}{%
          \let#1\ltx@undefined
          \directlua{%
            if tex.enableprimitives then %
              tex.enableprimitives('', {'\string#1'})%
            end%
          }%
          \ltx@ifundefined{\string#1}{%
          }{%
            \global#1=#1%
            \@PackageInfoNoLine{zref-pagelayout}{%
              \string#1 enabled%
            }%
          }%
        }{}%
      }%
      \ZREF@temp\pdfpagewidth
      \ZREF@temp\pdfpageheight
      \ZREF@temp\pdfhorigin
      \ZREF@temp\pdfvorigin
    \endgroup
  \fi
\fi
```

```
\def\ZREF@temp#1{%
  \begingroup
    \escapechar=-1 %
  \ltx@ifundefined{\string#1}{\endgroup}{%
    \edef\x{%
      \endgroup
      \noexpand\zref@newprop*{\string#1}%
                            [\number\noexpand#1]% hash-ok
                            {\number\noexpand#1}%
      \noexpand\zref@addprop{thepage}{\string#1}%
    }%
    \x
  }%
}
\ZREF@temp\mag
\ZREF@temp\paperwidth
\ZREF@temp\paperheight
\ZREF@temp\stockwidth
\ZREF@temp\stockheight
\ZREF@temp\pdfpagewidth
\ZREF@temp\pdfpageheight
\ZREF@temp\pdfhorigin
\ZREF@temp\pdfvorigin
\ZREF@temp\hoffset
\ZREF@temp\voffset
\ZREF@temp\topmargin
\ZREF@temp\oddsidemargin
\ZREF@temp\evensidemargin
\ZREF@temp\textwidth
\ZREF@temp\textheight
\ZREF@temp\headheight
\ZREF@temp\headsep
\ZREF@temp\footskip
\ZREF@temp\marginparwidth
\ZREF@temp\marginparsep
\ZREF@temp\columnwidth
\ZREF@temp\columnsep
```

\ifZREF@pl@list

```
\ltx@newif\ifZREF@pl@list
```

\zref@listpagelayout

```
\ZREF@IfDefinable\zlistpagelayout\def{%
  {\global\ZREF@pl@listtrue}%
}
```

\ZREF@pl@AfterLastShipout

```
\def\ZREF@pl@AfterLastShipout{%
  \ifZREF@pl@list
    \edef\ZREF@page@max{\the\value{abspage}}%
    \ltx@ifundefined{ZREF@org@testdef}{%
      \let\ZREF@org@testdef\@testdef
      \def\@testdef##1##2##3{%
        \ZREF@org@testdef{##1}{##2}{##3}%
        \def\ZREF@temp{##1}%
        \ifx\ZREF@temp\ZREF@RefPrefix
          \expandafter\gdef\csname##1@##2\endcsname{##3}%
        \fi
      }%
    }{}%
    \AtVeryEndDocument{\ZREF@pl@AtVeryEnd}%
  \fi
}
```

\ZREF@pl@AtVeryEnd

```
\def\ZREF@pl@AtVeryEnd{%
  \begingroup
    \toks@{Page layout parameters:\MessageBreak}%
    \count@=1 %
    \ZREF@pl@ListPage
    \edef\x{\endgroup
      \noexpand\@PackageInfoNoLine{zref-pagelayout}{\the\toks@}%
    }%
  \x
}
```

\ZREF@pl@ListPage

```
\def\ZREF@pl@ListPage{%
  \edef\x{%
    \toks@={%
      \the\toks@
      Page \the\count@:\noexpand\MessageBreak
      \zref@ifrefundefined{thepage\the\count@}{}{%
        \ltx@space\ltx@space mag = %
        \zref@extract{thepage\the\count@}{mag}%
        \noexpand\MessageBreak
        \ZREF@pl@ListEntry{paperwidth}%
        \ZREF@pl@ListEntry{paperheight}%
        \ZREF@pl@ListEntry{stockwidth}%
        \ZREF@pl@ListEntry{stockheight}%
        \ZREF@pl@ListEntry{pdfpagewidth}%
        \ZREF@pl@ListEntry{pdfpageheight}%
        \ZREF@pl@ListEntry{pdfhorigin}%
        \ZREF@pl@ListEntry{pdfvorigin}%
        \ZREF@pl@ListEntry{hoffset}%
        \ZREF@pl@ListEntry{voffset}%
        \ZREF@pl@ListEntry{topmargin}%
        \ZREF@pl@ListEntry{oddsidemargin}%
        \ZREF@pl@ListEntry{evensidemargin}%
        \ZREF@pl@ListEntry{textwidth}%
        \ZREF@pl@ListEntry{textheight}%
        \ZREF@pl@ListEntry{headheight}%
        \ZREF@pl@ListEntry{headsep}%
        \ZREF@pl@ListEntry{footskip}%
        \ZREF@pl@ListEntry{marginparwidth}%
        \ZREF@pl@ListEntry{marginparsep}%
        \ZREF@pl@ListEntry{columnwidth}%
        \ZREF@pl@ListEntry{columnsep}%
      }%
    }%
  }\x
  \ifnum\ZREF@page@max>\count@
    \advance\count@ by\ltx@one
  \else
    \expandafter\ltx@gobble
  \fi
  \ZREF@pl@ListPage
}
```

\ZREF@pl@ListEntry

```
\def\ZREF@pl@ListEntry#1{%
  \zref@ifpropundefined{#1}{%
  }{%
    \zref@ifrefcontainsprop{thepage\the\count@}{#1}{%
      \ltx@space\ltx@space#1 = %
      \zref@extract{thepage\the\count@}{#1}sp = %
      \the\dimexpr\zref@extract{thepage\the\count@}{#1}sp\relax
      \noexpand\MessageBreak
    }{}%
  }%
}
```

```
\AfterLastShipout{%
  \ZREF@pl@AfterLastShipout
}
```

⟨/pagelayout⟩

6.11 Module pageattr

```
⟨*pageattr⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-pageattr}%
  [2011/03/18 v2.21 Module pageattr for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi

\RequirePackage{ifluatex}[2010/03/01]

\ifluatex
  \ifnum\luatexversion<39 %
  \else
    \begingroup
      \escapechar=-1 %
      \def\ZREF@temp#1{%
        \ltx@IfUndefined{\string#1}{%
          \let#1\ltx@undefined
          \directlua{%
            if tex.enableprimitives then %
              tex.enableprimitives('', {'\string#1'})%
            end%
          }%
          \ltx@ifundefined{\string#1}{%
          }{%
            \global#1=#1%
            \@PackageInfoNoLine{zref-pageattr}{%
              \string#1 enabled%
            }%
          }%
        }{}%
      }%
      \ZREF@temp\pdfpageattr
      \ZREF@temp\pdfpagesattr
    \endgroup
  \fi
\fi

\let\ZREF@temp=N%
\ltx@IfUndefined{pdfpageattr}{%
  \@PackageInfoNoLine{zref-pageattr}{%
    \string\pdfpageattr\space is not available%
  }%
  \def\zref@pdfpageattr#1{}%
  \def\zref@pdfpageattr@used#1{}%
}{%
  \RequirePackage{zref-thepage}[2011/03/18]%
  \zref@newprop*{pdfpageattr}[]{\zref@hex{\the\pdfpageattr}}%
  \zref@addprop{thepage}{pdfpageattr}%
  \let\ZREF@temp=Y%
}
\ltx@IfUndefined{pdfpagesattr}{%
  \@PackageInfoNoLine{zref-pageattr}{%
    \string\pdfpagesattr\space is not available%
  }%
  \def\zref@pdfpagesattr{}%
  \def\zref@pdfpagesattr@used{}%
}{%
  \RequirePackage{zref-lastpage}[2011/03/18]%
  \zref@newprop*{pdfpagesattr}[]{\zref@hex{\the\pdfpagesattr}}%
  \zref@addprop{LastPage}{pdfpagesattr}%
  \let\ZREF@temp=Y%
}%
\ifx\ZREF@temp N%
  \expandafter\endinput
\fi
\RequirePackage{zref-abspage}[2011/03/18]
\RequirePackage{atveryend}[2010/03/24]
\RequirePackage{pdftexcmds}[2010/04/01]
\let\ZREF@temp=Y%
\ltx@IfUndefined{pdf@escapehex}{\let\ZREF@temp=N}{}
\ltx@IfUndefined{pdf@unescapehex}{\let\ZREF@temp=N}{}
\ifx\ZREF@temp N%
  \let\zref@hex\ltx@firstofone
  \let\zref@unhex\ltx@firstofone
\else
  \let\zref@hex\pdf@escapehex
  \let\zref@unhex\pdf@unescapehex
\fi
```

\ifZREF@pa@list

```
\ltx@newif\ifZREF@pa@list
```

\zref@listpageattr

```
\ZREF@IfDefinable\zlistpageattr\def{%
  {\ZREF@pa@listtrue}%
}
```

\ZREF@pa@AfterLastShipout

```
\def\ZREF@pa@AfterLastShipout{%
  \ifZREF@pa@list
    \edef\ZREF@page@max{\the\value{abspage}}%
    \ltx@ifundefined{ZREF@org@testdef}{%
      \let\ZREF@org@testdef\@testdef
      \def\@testdef##1##2##3{%
        \ZREF@org@testdef{##1}{##2}{##3}%
        \def\ZREF@temp{##1}%
        \ifx\ZREF@temp\ZREF@RefPrefix
          \expandafter\xdef\csname##1@##2\endcsname{##3}%
        \fi
      }%
    }{}%
    \AtVeryEndDocument{\ZREF@pa@AtVeryEnd}%
  \fi
}
```

\ZREF@pa@AtVeryEnd

```
\ltx@IfUndefined{pdfpageattr}{%
  \def\ZREF@pa@AtVeryEnd{}%
}{%
  \def\ZREF@pa@AtVeryEnd{%
    \begingroup
      \toks@{List of \ltx@backslashchar pdfpageattr:\MessageBreak}%
      \count@=1 %
      \ZREF@pa@ListPage
      \edef\x{\endgroup
        \noexpand\@PackageInfoNoLine{zref-pageattr}{%
          \the\toks@
        }%
      }%
    \x
  }%
```

\zref@pageattr

```
\def\zref@pageattr#1{%
  \zref@unhex{%
    \zref@extract{thepage\ZREF@number#1}{pdfpageattr}%
  }%
}
```

\zref@pageattr@used

```
\ZREF@Robust\def\zref@pageattr@used#1{%
  \zref@refused{thepage\ZREF@number#1}%
}
```

\ZREF@pa@ListPage

```
  \def\ZREF@pa@ListPage{%
    \edef\x{%
      \toks@={%
        \the\toks@
        Page \the\count@:%
        \noexpand\MessageBreak
        \zref@ifrefundefined{thepage\the\count@}{}{%
          <<\zref@pdfpageattr\count@>>%
          \noexpand\MessageBreak
        }%
      }%
    }\x
    \ifnum\ZREF@page@max>\count@
      \advance\count@ by\ltx@one
    \else
      \expandafter\ltx@gobble
    \fi
    \ZREF@pa@ListPage
  }%
}
```

```
\ltx@IfUndefined{pdfpagesattr}{%
}{%
```

\zref@pdfpagesattr

```
  \def\zref@pdfpagesattr{%
    \zref@unhex{%
      \zref@extract{LastPage}{pdfpagesattr}%
    }%
  }%
```

\zref@pdfpagesattr@used

```
  \ZREF@Robust\def\zref@pdfpagesattr@used{%
    \zref@refused{LastPage}%
  }%
```

```
  \ltx@LocalAppendToMacro\ZREF@pa@AtVeryEnd{%
    \@PackageInfoNoLine{zref-pageattr}{%
      \ltx@backslashchar pdfpagesattr:\MessageBreak
      <<\zref@pdfpagesattr>>%
      \MessageBreak
    }%
  }%
}
\AfterLastShipout{%
  \ZREF@pa@AfterLastShipout
}
```

```
⟨/pageattr⟩
```

6.12 Module marks

```
⟨*marks⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-marks}%
  [2011/03/18 v2.21 Module marks for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\newcommand*{\zref@marks@register}[3][]{%
  \edef\ZREF@TempName{#1}%
  \edef\ZREF@TempNum{\ZREF@number#2}%
  \ifnum\ZREF@TempNum<\ltx@zero %
    \PackageError\ZREF@name{%
      \string\zref@marks@register\ltx@space is called with invalid%
      \MessageBreak
      marks register number (\ZREF@TempNum)%
    }{%
      Use ‘0’ or the command, defined by \string\newmarks.\MessageBreak
      \@ehc
    }%
  \else
    \ifx\ZREF@TempName\ltx@empty
      \edef\ZREF@TempName{mark\romannumeral\ZREF@TempNum}%
    \else
      \edef\ZREF@TempName{marks\ZREF@TempName}%
    \fi
    \ZREF@MARKS@DefineProp{top}%
    \ZREF@MARKS@DefineProp{first}%
    \ZREF@MARKS@DefineProp{bot}%
    \kv@parse{#3}{%
      \ifx\kv@value\relax
        \def\kv@value{top,first,bot}%
      \fi
      \edef\ZREF@temp{\expandafter\ltx@car\kv@key X\@nil}%
      \ifx\ZREF@temp\ZREF@STAR
        \edef\kv@key{\expandafter\ltx@cdr\kv@key\@nil}%
        \zref@newlist\kv@key
      \fi
      \expandafter\comma@parse\expandafter{\kv@value}{%
        \ifcase0\ifx\comma@entry\ZREF@NAME@top 1\else
                \ifx\comma@entry\ZREF@NAME@first 1\else
                \ifx\comma@entry\ZREF@NAME@bot 1\fi\fi\fi\ltx@space
          \PackageWarning{zref-marks}{%
            Use ‘top’, ‘first’ or ‘bot’ for the list values%
            \MessageBreak
            in the third argument of \string\zref@marks@register.%
            \MessageBreak
            Ignoring unkown value ‘\comma@entry’%
          }%
        \else
          \zref@addprop{\kv@key}{\comma@entry\ZREF@TempName}%
        \fi
        \ltx@gobble
      }%
      \ltx@gobbletwo
    }%
  \fi
}
\def\ZREF@STAR{*}
\def\ZREF@NAME@top{top}
```

```
\def\ZREF@NAME@first{first}
\def\ZREF@NAME@bot{bot}
\def\ZREF@MARKS@DefineProp#1{%
  \zref@ifpropundefined{#1\ZREF@TempName}{%
    \ifnum\ZREF@TempNum=\ltx@zero
      \begingroup
        \edef\x{\endgroup
          \noexpand\zref@newprop*{#1\ZREF@TempName}[]{%
            \expandafter\noexpand\csname#1mark\endcsname
          }%
        }%
      \x
    \else
      \begingroup
        \edef\x{\endgroup
          \noexpand\zref@newprop*{#1\ZREF@TempName}[]{%
            \expandafter\noexpand\csname#1marks\endcsname
            \ZREF@TempNum
          }%
        }%
      \x
    \fi
  }{%
    \PackageWarning{zref-marks}{%
      \string\zref@marks@register\ltx@space does not generate the%
      \MessageBreak
      new property `#1\ZREF@TempName', because\MessageBreak
      it is already defined%
    }%
  }%
}
```

⟨/marks⟩

6.13 Module **runs**

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

⟨*runs⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-runs}%
  [2011/03/18 v2.21 Module runs for zref (HO)]%
```

\zruns

```
\providecommand*{\zruns}{0}%
\AtBeginDocument{%
  \edef\zruns{\number\numexpr\zruns+1}%
  \begingroup
    \def\on@line{}%
    \PackageInfo{zref-runs}{LaTeX runs: \zruns}%
    \if@filesw
      \immediate\write\@mainaux{%
        \string\gdef\string\zruns{\zruns}%
      }%
    \fi
  \endgroup
}
```

⟨/runs⟩

6.14 Module **perpage**

⟨*perpage⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-perpage}%
  [2011/03/18 v2.21 Module perpage for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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.

```
\RequirePackage{zref-abspage}[2011/03/18]
```

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

```
\zref@newprop*{pagevalue}[0]{\number\c@page}
\zref@newlist{perpage}
\zref@addprops{perpage}{abspage,page,pagevalue}
```

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

```
\newcounter{zpage}
```

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

```
\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`.

```
\let\ZREF@org@stepcounter\stepcounter
\def\stepcounter#1{%
  \ifcsname @stepcounterhook@#1\endcsname
    \csname @stepcounterhook@#1\endcsname
  \fi
  \ZREF@org@stepcounter{#1}%
}
```

`\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.

```
\ZREF@IfDefinable\zmakeperpage\def{%
  {%
    \@ifnextchar[\ZREF@makeperpage@opt{\ZREF@@makeperpage[\ltx@zero]}%
  }%
}
```

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

```
\def\ZREF@makeperpage@opt[#1]{%
  \begingroup
    \edef\x{\endgroup
      \noexpand\ZREF@@makeperpage[\number\numexpr#1-1\relax]%
    }%
  \x
}
```

```
\def\ZREF@@makeperpage[#1]#2{%
  \@ifundefined{@stepcounterhook@#2}{%
    \expandafter\gdef\csname @stepcounterhook@#2\endcsname{}%
  }{}%
  \expandafter\gdef\csname ZREF@perpage@#2\endcsname{%
    \ZREF@@perpage@step{#2}{#1}%
  }%
  \expandafter\g@addto@macro\csname @stepcounterhook@#2\endcsname{%
    \ifcsname ZREF@perpage@#2\endcsname
      \csname ZREF@perpage@#2\endcsname
    \fi
  }%
}
```

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

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

First the reference is generated.

```
  \global\advance\c@zref@unique\ltx@one
  \begingroup
    \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 kwown, we use the page counter as approximation.

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

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

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

Page changes are detected by a changed absolute page number.

```
    \expandafter\ifx\csname ZREF@abspage@#1\expandafter\endcsname
                    \csname ZREF@currentabspage@#1\endcsname
    \else
      \global\csname c@#1\endcsname=#2\relax
      \global\expandafter\let
          \csname ZREF@currentabspage@#1\expandafter\endcsname
          \csname ZREF@abspage@#1\endcsname
    \fi
  \endgroup
}
```

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

```
\ZREF@IfDefinable\zunmakeperpage\def{%
  #1{%
    \global\expandafter
    \let\csname ZREF@perpage@#1\endcsname\@undefined
  }%
}
```

```
⟨/perpage⟩
```

6.15 Module **titleref**

```
⟨*titleref⟩
\NeedsTeXFormat{LaTeX2e}
```

```
\ProvidesPackage{zref-titleref}%
  [2011/03/18 v2.21 Module titleref for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
\RequirePackage{gettitlestring}[2009/12/08]
```

6.15.1 Implementation

```
\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.

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

```
\ZREF@NewPropTitle
\zref@addprop\ZREF@mainlist{title}%
```

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

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

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

```
\newif\ifzref@titleref@expand
```

`\ZREF@titleref@hook` The hook `\ZREF@titleref@hook` allows to extend the cleanup for the expansion method. Thus unnecessary macros can be removed or dangerous commands removed. The hook is executed before the expansion of `\zref@titleref@current`.

```
\let\ZREF@titleref@hook\ltx@empty
```

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

```
\ZREF@Robust\def\zref@titleref@cleanup#1{%
  \begingroup
    \toks@\expandafter{%
      \ZREF@titleref@hook
      #1%
    }%
  \expandafter\endgroup
  \expandafter\def\expandafter\ZREF@titleref@hook\expandafter{%
    \the\toks@
  }%
}%
```

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

```
\newif\ifzref@titleref@stripperiod
\zref@titleref@stripperiodtrue
```

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

```
\ZREF@Robust\def\zref@titleref@setcurrent#1{%
  \ifzref@titleref@expand
```

```
    \GetTitleStringExpand{#1}%
  \else
    \GetTitleStringNonExpand{#1}%
  \fi
  \edef\zref@titleref@current{%
    \detokenize\expandafter{\GetTitleStringResult}%
  }%
  \ifzref@titleref@stripperiod
    \edef\zref@titleref@current{%
      \expandafter\ZREF@stripperiod\zref@titleref@current
      \ltx@empty.\ltx@empty\@nil
    }%
  \fi
}%
\GetTitleStringDisableCommands{%
  \ZREF@titleref@hook
}
```

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

```
\def\ZREF@stripperiod#1.\ltx@empty#2\@nil{#1}%
```

6.15.2 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`.

```
\define@key{ZREF@TR}{expand}[true]{%
  \csname zref@titleref@expand#1\endcsname
}%
\define@key{ZREF@TR}{stripperiod}[true]{%
  \csname zref@titleref@stripperiod#1\endcsname
}%
\define@key{ZREF@TR}{cleanup}{%
  \zref@titleref@cleanup{#1}%
}%
\define@key{ZREF@TR}{title}{%
  \def\zref@titleref@current{#1}%
}%
\ZREF@IfDefinable\ztitlerefsetup\def{%
  {\kvsetkeys{ZREF@TR}}%
}%
```

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

```
\ZREF@IfDefinable\ztitleref\def{%
  {\zref@wrapper@babel\ZREF@titleref}%
}%
\def\ZREF@titleref#1{%
  \begingroup
    \zref@refused{#1}%
    \let\label\ltx@gobble
    \zref@extract{#1}{title}%
  \endgroup
}%
```

6.15.3 Patches for section and caption commands

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

Captions of figures and tables.

```
\AtBeginDocument{%
```

```
\ZREF@patch{@caption}{%
  \long\def\@caption#1[#2]{%
    \zref@titleref@setcurrent{#2}%
    \ZREF@org@@caption{#1}[{#2}]%
  }%
}%
```

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

```
\ZREF@patch{@part}{%
  \def\@part[#1]{%
    \zref@titleref@setcurrent{#1}%
    \ZREF@org@@part[{#1}]%
  }%
}%
\ZREF@patch{@chapter}{%
  \def\@chapter[#1]{%
    \zref@titleref@setcurrent{#1}%
    \ZREF@org@@chapter[{#1}]%
  }%
}%
\ZREF@patch{@sect}{%
  \def\@sect#1#2#3#4#5#6[#7]{%
    \zref@titleref@setcurrent{#7}%
    \ZREF@org@@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}]%
  }%
}%
```

The star versions of the section commands.

```
\ZREF@patch{@spart}{%
  \def\@spart#1{%
    \zref@titleref@setcurrent{#1}%
    \ZREF@org@@spart{#1}%
  }%
}%
\ZREF@patch{@schapter}{%
  \def\@schapter#1{%
    \zref@titleref@setcurrent{#1}%
    \ZREF@org@@schapter{#1}%
  }%
}%
\ZREF@patch{@ssect}{%
  \def\@ssect#1#2#3#4#5{%
    \zref@titleref@setcurrent{#5}%
    \ZREF@org@@ssect{#1}{#2}{#3}{#4}{#5}%
  }%
}%
```

6.15.4 Environment **description**

```
\ZREF@patch{descriptionlabel}{%
  \def\descriptionlabel#1{%
    \zref@titleref@setcurrent{#1}%
    \ZREF@org@descriptionlabel{#1}%
  }%
}%
```

6.15.5 Class **memoir**

```
\@ifclassloaded{memoir}{%
  \ltx@IfUndefined{ifheadnameref}{}{%
    \def\@chapter[#1]#2{%
      \ltx@IfUndefined{ch@pt@c}{%
        \zref@titleref@setcurrent{#1}%
      }{%
        \ifx\ch@pt@c\ltx@empty
          \zref@titleref@setcurrent{#2}%
        \else
          \def\NR@temp{#1}%
          \ifx\NR@temp\ltx@empty
            \expandafter\zref@titleref@setcurrent
            \expandafter{\ch@pt@c}%
          \else
            \ifheadnameref
              \zref@titleref@setcurrent{#1}%
            \else
              \expandafter\zref@titleref@setcurrent
              \expandafter{\ch@pt@c}%
            \fi
          \fi
        \fi
      }%
      \ZREF@org@@chapter[{#1}]{#2}%
    }%
    \ZREF@patch{M@sect}{%
      \def\M@sect#1#2#3#4#5#6[#7][#8]{%
        \ifheadnameref
          \zref@titleref@setcurrent{#8}%
        \else
          \zref@titleref@setcurrent{#7}%
        \fi
        \ZREF@org@M@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}][{#8}]%
      }%
    }%
  }%
}{}%
```

6.15.6 Class **beamer**

```
\@ifclassloaded{beamer}{%
  \ZREF@patch{beamer@section}{%
    \long\def\beamer@section[#1]{%
      \zref@titleref@setcurrent{#1}%
      \ZREF@org@beamer@section[{#1}]%
    }%
  }%
  \ZREF@patch{beamer@subsection}{%
    \long\def\beamer@subsection[#1]{%
      \zref@titleref@setcurrent{#1}%
      \ZREF@org@beamer@subsection[{#1}]%
    }%
  }%
  \ZREF@patch{beamer@subsubsection}{%
    \long\def\beamer@subsubsection[#1]{%
      \zref@titleref@setcurrent{#1}%
      \ZREF@org@beamer@subsubsection[{#1}]%
    }%
  }%
}{}%
```

6.15.7 Package **titlesec**

```
\@ifpackageloaded{titlesec}{%
  \ZREF@patch{ttl@sect@i}{%
    \def\ttl@sect@i#1#2[#3]#4{%
      \zref@titleref@setcurrent{#4}%
      \ZREF@org@ttl@sect@i{#1}{#2}[{#3}]{#4}%
    }%
  }%
}{}%
```

6.15.8 Package longtable

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

```
\@ifpackageloaded{longtable}{%
  \ZREF@patch{LT@c@ption}{%
    \def\LT@c@ption#1[#2]#3{%
      \ZREF@org@LT@c@ption{#1}[{#2}]{#3}%
      \zref@titleref@setcurrent{#2}%
    }%
  }%
}{}%
```

6.15.9 Package listings

Package listings: support for its caption.

```
\@ifpackageloaded{listings}{%
  \ZREF@patch{lst@MakeCaption}{%
    \def\lst@MakeCaption{%
      \ifx\lst@label\ltx@empty
      \else
        \expandafter\zref@titleref@setcurrent\expandafter{%
          \lst@@caption
        }%
      \fi
      \ZREF@org@lst@MakeCaption
    }%
  }%
}{}%
```

6.15.10 Theorems

```
\ZREF@patch{@opargbegintheorem}{%
  \def\@opargbegintheorem#1#2#3{%
    \zref@titleref@setcurrent{#3}%
    \ZREF@org@@opargbegintheorem{#1}{#2}{#3}%
  }%
}%
\@ifpackageloaded{amsthm}{%
  \begingroup
    \edef\x{macro:\string#1\string#2[\string#3]}%
    \@onelevel@sanitize\x
    \def\y#1->#2\@nil{#1}%
    \edef\z{\expandafter\y\meaning\@begintheorem->\@nil}%
    \@onelevel@sanitize\z
  \expandafter\endgroup
  \ifx\x\z
    \ZREF@patch{@begintheorem}{%
      \def\@begintheorem#1#2[#3]{%
        \zref@titleref@setcurrent{#3}%
        \ZREF@org@@begintheorem{#1}{#2}[{#3}]%
      }%
    }%
  \fi
}{}%
```

1922 }

1923 ⟨/titleref⟩

6.16 Module xr

1924 ⟨*xr⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-xr}%
  [2011/03/18 v2.21 Module xr for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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

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

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

```
\zref@newprop{externaldocument}{}%
```

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

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

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

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

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

```
\SetupKeyvalOptions{%
  family=ZREF@XR,%
  prefix=ZREF@xr@%
}
\DeclareBoolOption[true]{tozreflabel}
\DeclareBoolOption[false]{toltxlabel}
\DeclareBoolOption{verbose}
\define@key{ZREF@XR}{ext}{%
  \def\zref@xr@{#1}%
}
\DeclareBoolOption[false]{urluse}
```

`\zxrsetup`

```
\newcommand*{\zxrsetup}{%
  \kvsetkeys{ZREF@XR}%
}%
```

`\ZREF@xr@URL`

```
\newcount\ZREF@xr@URL
\ZREF@xr@URL=\ltx@zero
```

`\ZREF@xr@AddURL`

```
\def\ZREF@xr@AddURL#1{%
  \begingroup
    \def\ZREF@temp{#1}%
    \count@=\ltx@one
    \ZREF@xr@@AddUrl
  \endgroup
}
```

\ZREF@xr@@AddUrl

```
\def\ZREF@xr@@AddUrl{%
  \ifnum\count@>\ZREF@xr@URL
    \global\advance\ZREF@xr@URL by\ltx@one
    \xdef\ZREF@xr@theURL{\romannumeral\ZREF@xr@URL}%
    \global\expandafter\let
        \csname Z@U@\ZREF@xr@theURL\endcsname\ZREF@temp
    \@PackageInfo{zref-xr}{%
      \ltx@backslashchar Z@U@\ZREF@xr@theURL:\MessageBreak
      \ZREF@temp\MessageBreak
    }%
  \else
    \expandafter
    \ifx\csname Z@U@\romannumeral\count@\endcsname\ZREF@temp
      \xdef\ZREF@xr@theURL{\romannumeral\count@}%
    \else
      \expandafter\expandafter\expandafter\ZREF@xr@@AddUrl
    \fi
  \fi
}
```

\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).

```
\ZREF@IfDefinable\zexternaldocument\def{%
  {%
    \ZREF@NewPropAnchor
    \ZREF@NewPropTitle
    \begingroup
      \csname @safe@actives@true\endcsname
      \makeatletter
      \@ifstar{%
        \ZREF@xr@zreflabelfalse
        \@testopt\ZREF@xr@externaldocument{}%
      }{%
        \ZREF@xr@zreflabeltrue
        \@testopt\ZREF@xr@externaldocument{}%
      }%
  }%
}%
```

If the `\include` featuer 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 LaTeX's order using `\input`.

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

```
\def\ZREF@xr@externaldocument[#1]#2{%
    \def\ZREF@xr@prefix{#1}%
    \let\ZREF@xr@filelist\ltx@empty
    \edef\ZREF@xr@externalfile{#2}%
    \edef\ZREF@xr@file{\ZREF@xr@externalfile.aux}%
    \filename@parse{#2}%
    \@testopt\ZREF@xr@graburl{#2.\zref@xr@ext}%
}%
\def\ZREF@xr@graburl[#1]{%
    \edef\ZREF@xr@url{#1}%
    \ifZREF@xr@urluse
      \expandafter\ZREF@xr@AddURL\expandafter{\ZREF@xr@url}%
      \expandafter\def\expandafter\ZREF@xr@url
      \expandafter{\csname Z@U@\ZREF@xr@theURL\endcsname}%
    \fi
    \ZREF@xr@checkfile
  \endgroup
}%
```

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

```
\def\ZREF@xr@checkfile{%
  \openin\@inputcheck\ZREF@xr@file\relax
  \ifeof\@inputcheck
    \PackageWarning{zref-xr}{%
      File `\ZREF@xr@file' not found or empty,\MessageBreak
      labels not imported%
    }%
  \else
    \PackageInfo{zref-xr}{%
      Label \ifZREF@xr@zreflabel (zref) \fi
      import from `\ZREF@xr@file'%
    }%
    \def\ZREF@xr@found{0}%
    \def\ZREF@xr@ignored@empty{0}%
    \def\ZREF@xr@ignored@zref{0}%
    \def\ZREF@xr@ignored@ltx{0}%
    \ZREF@xr@processfile
    \closein\@inputcheck
    \begingroup
      \let\on@line\ltx@empty
      \PackageInfo{zref-xr}{%
        Statistics for `\ZREF@xr@file':\MessageBreak
        \ZREF@xr@found\space
        \ifZREF@xr@zreflabel zref\else LaTeX\fi\space
        label(s) found%
        \ifnum\ZREF@xr@ignored@empty>0 %
          ,\MessageBreak
          \ZREF@xr@ignored@empty\space empty label(s) ignored%
        \fi
        \ifnum\ZREF@xr@ignored@zref>0 %
          ,\MessageBreak
          \ZREF@xr@ignored@zref\space
          duplicated zref label(s) ignored%
        \fi
        \ifnum\ZREF@xr@ignored@ltx>0 %
          ,\MessageBreak
          \ZREF@xr@ignored@ltx\space
          duplicated latex label(s) ignored%
        \fi
      }%
    \endgroup
  \fi
  \ifx\ZREF@xr@filelist\ltx@empty
  \else
    \edef\ZREF@xr@file{%
      \expandafter\ltx@car\ZREF@xr@filelist\@nil
    }%
    \edef\ZREF@xr@filelist{%
      \expandafter\ltx@cdr\ZREF@xr@filelist\ltx@empty\@nil
    }%
    \expandafter\ZREF@xr@checkfile
  \fi
}%
```

`\ZREF@xr@processfile`

```
\def\ZREF@xr@processfile{%
```

```
  \read\@inputcheck to\ZREF@xr@line
  \expandafter\ZREF@xr@processline\ZREF@xr@line..\ZREF@nil
  \ifeof\@inputcheck
  \else
    \expandafter\ZREF@xr@processfile
  \fi
}%
```

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

```
\long\def\ZREF@xr@processline#1#2#3\ZREF@nil{%
  \def\x{#1}%
  \toks@{#2}%
  \ifZREF@xr@zreflabel
    \ifx\x\ZREF@xr@zref@newlabel
      \expandafter
      \ZREF@xr@process@zreflabel\ZREF@xr@line...\ZREF@nil
    \fi
  \else
    \ifx\x\ZREF@xr@newlabel
      \expandafter
      \ZREF@xr@process@label\ZREF@xr@line...[]\ZREF@nil
    \fi
  \fi
  \ifx\x\ZREF@xr@@input
    \edef\ZREF@xr@filelist{%
      \etex@unexpanded\expandafter{\ZREF@xr@filelist}%
      {\filename@area\the\toks@}%
    }%
  \fi
}%
\def\ZREF@xr@process@zreflabel\zref@newlabel#1#2#3\ZREF@nil{%
  \edef\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
  \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
  \def\x{#2}%
  \edef\ZREF@xr@tempname{$temp$}%
  \edef\ZREF@xr@temprefname{Z@R@\ZREF@xr@tempname}%
  \let\ZREF@xr@list\x
  \ifx\ZREF@xr@list\ltx@empty
    \PackageWarningNoLine{zref-xr}{%
      Label `#1' without properties ignored\MessageBreak
      in file `\ZREF@xr@file'%
    }%
    \edef\ZREF@xr@ignored@empty{%
      \the\numexpr\ZREF@xr@ignored@empty+1\relax
    }%
  \else
    \expandafter\ZREF@xr@checklist\x\ZREF@nil
    \expandafter\let\csname\ZREF@xr@temprefname\endcsname\x
    \expandafter\ltx@LocalAppendToMacro
    \csname\ZREF@xr@temprefname\expandafter\endcsname
    \expandafter{%
      \expandafter\externaldocument\expandafter{%
        \ZREF@xr@externalfile
      }%
    }%
    \ZREF@xr@urlcheck\ZREF@xr@tempname
    \ifZREF@xr@tozreflabel
      \@ifundefined{\ZREF@xr@refname}{%
        \ifZREF@xr@verbose
          \PackageInfo{zref-xr}{%
            Import to zref label `\ZREF@xr@tempname#1'%
          }%
        \fi
        \global\expandafter
        \let\csname\ZREF@xr@refname\expandafter\endcsname
        \csname\ZREF@xr@temprefname\endcsname
      }{%
        \ZREF@xr@zref@ignorewarning{\ZREF@xr@prefix#1}%
      }%
    \fi
    \ifZREF@xr@toltxlabel
      \ZREF@xr@tolabel{\ZREF@xr@tempname}{\ZREF@xr@prefix#1}%
    \fi
  \fi
}%
\def\ZREF@xr@process@label\newlabel#1#2#3[#4]#5\ZREF@nil{%
  \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
  \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
  \def\x{#2}%
  \edef\ZREF@xr@tempname{$temp$}%
  \edef\ZREF@xr@temprefname{Z@R@\ZREF@xr@tempname}%
  \expandafter\ZREF@xr@scanparams
      \csname\ZREF@xr@temprefname\expandafter\endcsname
      \x{}{}{}{}{}\ZREF@nil
  \ifx\\#4\\%
  \else
    % ntheorem knows an optional argument at the end of \newlabel
    \ZREF@NewPropTheotype
    \expandafter\ltx@LocalAppendToMacro
        \csname\ZREF@xr@temprefname\endcsname{\theotype{#4}}%
  \fi
  \expandafter\ltx@LocalAppendToMacro
  \csname\ZREF@xr@temprefname\expandafter\endcsname\expandafter{%
    \expandafter\externaldocument\expandafter{%
      \ZREF@xr@externalfile
    }%
  }%
  \ZREF@xr@urlcheck\ZREF@xr@tempname
  \ifZREF@xr@tozreflabel
    \@ifundefined{\ZREF@xr@refname}{%
      \ifZREF@xr@verbose
        \PackageInfo{zref-xr}{%
          Import to zref label `\ZREF@xr@prefix#1'%
        }%
      \fi
      \global\expandafter
      \let\csname\ZREF@xr@refname\expandafter\endcsname
      \csname\ZREF@xr@temprefname\endcsname
    }{%
      \ZREF@xr@zref@ignorewarning{\ZREF@xr@prefix#1}%
    }%
  \fi
  \ifZREF@xr@toltxlabel
    \ZREF@xr@tolabel{\ZREF@xr@tempname}{\ZREF@xr@prefix#1}%
  \fi
}
\def\ZREF@xr@zref@newlabel{\zref@newlabel}%
\def\ZREF@xr@newlabel{\newlabel}%
\def\ZREF@xr@@input{\@input}%
\def\ZREF@xr@relax{\relax}%
```

\ZREF@xr@tolabel

```
\def\ZREF@xr@tolabel#1#2{%
  \ifZREF@xr@verbose
    \PackageInfo{zref-xr}{%
      Import to LaTeX label `#2'%
    }%
  \fi
  \zref@wrapper@unexpanded{%
    \expandafter\xdef\csname r@#2\endcsname{%
      {%
        \ltx@ifundefined{M@TitleReference}{%
          \ltx@ifundefined{TR@TitleReference}{%
            \zref@extractdefault{#1}{default}{}%
          }{%
            \noexpand\TR@TitleReference
            {\zref@extractdefault{#1}{default}{}}%
            {\zref@extractdefault{#1}{title}{}}%
          }%
        }{%
          \noexpand\M@TitleReference
          {\zref@extractdefault{#1}{default}{}}%
          {\zref@extractdefault{#1}{title}{}}%
        }%
      }%
      {\zref@extractdefault{#1}{page}{}}%
      \ltx@ifpackageloaded{nameref}{%
        {\zref@extractdefault{#1}{title}{}}%
        {\zref@extractdefault{#1}{anchor}{}}%
        \zref@ifrefcontainsprop{#1}{urluse}{%
          {\zref@extractdefault{#1}{urluse}{}}%
        }{%
          {\zref@extractdefault{#1}{url}{}}%
        }%
      }{}%
    }%
  }%
}
```

\ZREF@xr@zref@ignorewarning

```
\def\ZREF@xr@zref@ignorewarning#1{%
  \PackageWarningNoLine{zref-xr}{%
    Zref label ‘#1’ is already in use\MessageBreak
    in file ‘\ZREF@xr@file’%
  }%
  \edef\ZREF@xr@ignored@zref{%
    \the\numexpr\ZREF@xr@ignored@zref+1%
  }%
}%
```

\ZREF@xr@ltx@ignorewarning

```
\def\ZREF@xr@ltx@ignorewarning#1{%
  \PackageWarningNoLine{zref-xr}{%
    LaTeX label ‘#1’ is already in use\MessageBreak
    in file ‘\ZREF@xr@file’%
  }%
  \edef\ZREF@xr@ignored@ltx{%
    \the\numexpr\ZREF@xr@ignored@ltx+1%
  }%
}%
```

\ZREF@xr@checklist

```
\def\ZREF@xr@checklist#1#2#3\ZREF@nil{%
  \ifx\@undefined#1\relax
    \expandafter\ZREF@xr@checkkey\string#1\@nil
  \fi
  \ifx\\#3\\%
  \else
```

```
    \ltx@ReturnAfterFi{%
      \ZREF@xr@checklist#3\ZREF@nil
    }%
  \fi
}%
\def\ZREF@xr@checkkey#1#2\@nil{%
  \zref@ifpropundefined{#2}{%
    \zref@newprop{#2}{}%
  }{}%
}%
```

\ZREF@xr@scanparams

```
\def\ZREF@xr@scanparams#1#2#3#4#5#6#7\ZREF@nil{%
  \let#1\ltx@empty
  \ZREF@foundfalse
  \ZREF@xr@scantitleref#1#2\TR@TitleReference{}{}\ZREF@nil
  \ifZREF@found
  \else
    \ltx@LocalAppendToMacro#1{\default{#2}}%
  \fi
  % page
  \ltx@LocalAppendToMacro#1{\page{#3}}%
  % nameref title
  \ifZREF@found
  \else
    \ifx\\#4\\%
    \else
      \def\ZREF@xr@temp{#4}%
      \ifx\ZREF@xr@temp\ZREF@xr@relax
      \else
        \ltx@LocalAppendToMacro#1{\title{#4}}%
      \fi
    \fi
  \fi
  % anchor
  \ifx\\#5\\%
  \else
    \ltx@LocalAppendToMacro#1{\anchor{#5}}%
  \fi
  \ifx\\#6\\%
  \else
    \ifZREF@xr@urluse
      \ZREF@xr@AddURL{#6}%
      \expandafter\ltx@LocalAppendToMacro\expandafter#1%
      \expandafter{%
        \expandafter\urluse\expandafter{%
          \csname Z@U@\ZREF@xr@theURL\endcsname
        }%
      }%
    \else
      \ltx@LocalAppendToMacro#1{\url{#6}}%
    \fi
  \fi
}%
```

\ZREF@xr@scantitleref

```
\def\ZREF@xr@scantitleref#1#2\TR@TitleReference#3#4#5\ZREF@nil{%
  \ifx\\#5\\%
  \else
    \ltx@LocalAppendToMacro#1{%
      \default{#3}%
      \title{#4}%
    }%
    \ZREF@foundtrue
  \fi
}%
```

\ZREF@xr@urlcheck

```
\def\ZREF@xr@urlcheck#1{%
  \zref@ifrefcontainsprop{#1}{anchor}{%
    \zref@ifrefcontainsprop{#1}{url}{%
    }{%
      \expandafter
      \ltx@LocalAppendToMacro\csname Z@R@#1\expandafter\endcsname
      \expandafter{%
        \csname url\ifZREF@xr@urluse use\fi
        \expandafter\endcsname\expandafter{\ZREF@xr@url}%
      }%
    }%
  }{%
  }%
}%
```

⟨/xr⟩

6.17 Module **hyperref**

UNFINISHED :-(

⟨*hyperref⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-hyperref}%
  [2011/03/18 v2.21 Module hyperref for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\ZREF@NewPropAnchor
\zref@addprop\ZREF@mainlist{anchor}%
```

⟨/hyperref⟩

6.18 Module **savepos**

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

6.18.1 Identification

⟨*savepos⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-savepos}%
  [2011/03/18 v2.21 Module savepos for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

6.18.2 Availability

First we check, whether the feature is available.

```
\ltx@IfUndefined{pdfsavepos}{%
  \PackageError\ZREF@name{%
```

```
    \string\pdfsavepos\space is not supported.\MessageBreak
    It is provided by pdfTeX (1.40) or XeTeX%
  }\ZREF@UpdatePdfTeX
  \endinput
}{}%
```

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`.

```
\RequirePackage{ifpdf}
\ifpdf
\else
  \ltx@IfUndefined{pdftexversion}{%
  }{%
    \ifnum\pdftexversion<140 %
      \PackageError\ZREF@name{%
        \string\pdfsavepos\space is not supported in DVI mode%
        \MessageBreak
        of this pdfTeX version%
      }\ZREF@UpdatePdfTeX
      \expandafter\expandafter\expandafter\endinput
    \fi
  }%
\fi
```

6.18.3 Setup

```
\zref@newlist{savepos}
\zref@newprop*{posx}[0]{\the\pdflastxpos}
\zref@newprop*{posy}[0]{\the\pdflastypos}
\zref@addprops{savepos}{posx,posy}
```

6.18.4 User macros

`\zref@savepos`

```
\def\zref@savepos{%
  \if@filesw
    \pdfsavepos
  \fi
}
```

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

```
\ZREF@IfDefinable\zsavepos\def{%
  #1{%
    \@bsphack
    \if@filesw
      \zref@savepos
      \zref@labelbylist{#1}{savepos}%
    \fi
    \@esphack
  }%
}
```

`\zposx` `\zposy` 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.

```
\newcommand*{\zposx}[1]{%
  \zref@extract{#1}{posx}%
}%
\newcommand*{\zposy}[1]{%
  \zref@extract{#1}{posy}%
}%
```

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

Also it is in the responsibility of the user to marked used positions by `\zrefused` in order to notify LaTeX about undefined references.

`\ZREF@savepos@ok`

```
\let\ZREF@savepos@ok=Y
```

```
⟨/savepos⟩
```

6.19 Module **abspos**

6.19.1 Identification

```
⟨*abspos⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-abspos}%
  [2011/03/18 v2.21 Module abspos for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\RequirePackage{zref-savepos}[2011/03/18]
\ifx\ZREF@savepos@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\RequirePackage{zref-pagelayout}[2011/03/18]
\zref@addprop{savepos}{abspage}
```

```
⟨/abspos⟩
```

6.20 Module **dotfill**

```
⟨*dotfill⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-dotfill}%
  [2011/03/18 v2.21 Module dotfill for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

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

```
\RequirePackage{zref-savepos}[2011/03/18]
```

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

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

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

```
\RequirePackage{keyval}
```

The definitions of the keys follow.

```
\define@key{ZREF@DF}{unit}{%
  \def\ZREF@df@unit{#1}%
}
\define@key{ZREF@DF}{min}{%
  \def\ZREF@df@min{#1}%
}
\define@key{ZREF@DF}{dot}{%
```

```
  \def\ZREF@df@dot{#1}%
}
```

Defaults are set, see user interface.

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

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

```
\newcommand*{\zdotfillsetup}{\kvsetkeys{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.

```
\ZREF@IfDefinable\zdotfill\def{%
  {%
    \leavevmode
    \global\advance\c@zref@unique\ltx@one
    \begingroup
      \def\ZREF@temp{zref@\number\c@zref@unique}%
      \pdfsavepos
      \zref@labelbyprops{\thezref@unique L}{posx}%
      \setlength{\dimen@}{\ZREF@df@unit}%
      \zref@ifrefundefined{\thezref@unique R}{%
        \ZREF@dotfill
      }{%
        \ifnum\numexpr\zposx{\thezref@unique R}%
                     -\zposx{\thezref@unique L}\relax
            <\dimexpr\ZREF@df@min\dimen@\relax
          \hfill
        \else
          \ZREF@dotfill
        \fi
      }%
      \pdfsavepos
      \zref@labelbyprops{\thezref@unique R}{posx}%
    \endgroup
    \kern\z@
  }%
}
```

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

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

```
⟨/dotfill⟩
```

6.21 Module **env**

```
⟨*env⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zref-env}%
  [2011/03/18 v2.21 Module env for zref (HO)]%
\RequirePackage{zref-base}[2011/03/18]
\ifx\ZREF@base@ok Y%
\else
  \expandafter\endinput
\fi
```

```
\zref@newprop{envname}[]{\@currenvir}
\zref@newprop{envline}[]{\zref@env@line}
```

`\zref@env@line` Macro `\zref@env@line` extracts the line number from `\@currenvline`.

```
\def\zref@env@line{%
```

```
  \ifx\@currenvline\ltx@empty
  \else
    \expandafter
    \ZREF@ENV@line\@currenvline\ltx@empty line \ltx@empty\@nil
  \fi
}
```

\ZREF@ENV@line

```
\def\ZREF@ENV@line#1line #2\ltx@empty#3\@nil{#2}%
```

⟨/env⟩

7 Test

7.1 \zref@localaddprop

```
⟨*test1⟩
\NeedsTeXFormat{LaTeX2e}
\nofiles
\documentclass{article}
\usepackage{zref-base}[2011/03/18]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}

\makeatletter
\def\ExpectList#1#2{%
  \expandafter\expandafter\expandafter\Expect
  \expandafter\expandafter\expandafter{\csname Z@L@#1\endcsname}{#2}%
}
\begin{qstest}{localaddprop}{localaddprop}
  \ExpectList{main}{\default\page}%
  \Expect{undefined}*{\meaning\foobar}%
  \zref@newprop{foobar}{FOO}%
  \Expect{undefined}*{\meaning\foobar}%
  \zref@newlist{alist}%
  \ExpectList{alist}{}%
  \begingroup
    \zref@localaddprop{main}{foobar}%
    \Expect{undefined}*{\meaning\foobar}%
    \ExpectList{main}{\default\page\foobar}%
    \zref@localaddprop{alist}{page}%
    \ExpectList{alist}{\page}%
  \endgroup
  \ExpectList{main}{\default\page}%
  \ExpectList{alist}{}%
  \zref@addprop{alist}{foobar}%
  \ExpectList{alist}{\foobar}%
  \Expect{undefined}*{\meaning\foobar}%
\end{qstest}
\@@end
⟨/test1⟩
```

7.2 Module **base**

```
⟨*test-base⟩
\NeedsTeXFormat{LaTeX2e}
\documentclass{article}
\usepackage{zref-base,zref-titleref}[2011/03/18]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}

```

```
\makeatletter
\newcommand*{\DefExpand}[2]{%
  \expandafter\expandafter\expandafter\def
  \expandafter\expandafter\expandafter#1%
  \expandafter\expandafter\expandafter{#2}%
  \@onelevel@sanitize#1%
}
\newcommand*{\Test}[3]{%
  \Expect{#2}*{#1}%
  \zref@wrapper@unexpanded{%
    \Expect*{#3}*{#1}%
  }%
  \DefExpand\x{#1}%
  \Expect*{#3}*{\x}%
}
\makeatother

\begin{document}
\section{\textit{Hello} \textbf{World}}
\label{sec:hello}
\makeatletter
\zref@newprop{foo}[\@empty D\@empty efault]{\@empty V\@empty alue}
\begin{qstest}{getcurrent}{getcurrent}
  \Test{\zref@getcurrent{foo}}%
       {Value}{\noexpand\@empty V\noexpand\@empty alue}%
  \Test{\zref@getcurrent{xy}}{}{}%
\end{qstest}
\begin{qstest}{extract}{extract}
  \def\textbf#1{<#1>}%
  \def\textit#1{[#1]}% hash-ok
  \Test{\zref@extractdefault{xy}{page}{\@empty D\@empty efault}}%
       {Default}{\noexpand\@empty D\noexpand\@empty efault}%
  \Test{\zref@extractdefault{sec:hello}{foo}{\@empty A\@empty B}}%
       {AB}{\noexpand\@empty A\noexpand\@empty B}%
  \Test{\zref@extract{sec:hello}{foo}}%
       {Default}{\noexpand\@empty D\noexpand\@empty efault}%
  \zref@ifrefundefined{sec:hello}{%
  }{%
    \Test{\zref@extract{sec:hello}{default}}{1}{1}%
    \Test{\zref@extract{sec:hello}{title}}%
         {[Hello] <World>}%
         {\noexpand\textit{Hello} \noexpand\textbf{World}}%
  }%
\end{qstest}
\end{document}
⟨/test-base⟩
```

7.3 Module runs

```
⟨*test-runs⟩
\NeedsTeXFormat{LaTeX2e}
\documentclass{article}
\usepackage{zref-runs}[2011/03/18]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}

\begin{qstest}{zruns-preamble}{zruns-preamble}
  \Expect{0}*{\zruns}%
\end{qstest}

\AtBeginDocument{%
  \begin{qstest}{zruns-atbegindocument}{zruns-atbegindocument}%
```

```
    \Expect*{\number\ExpectRuns}*{\zruns}%
  \end{qstest}%
}

\begin{document}
\begin{qstest}{zruns-document}{zruns-document}
  \Expect*{\number\ExpectRuns}*{\zruns}%
\end{qstest}
\end{document}
⟨/test-runs⟩
```

7.4 Module **titleref**

```
⟨*test-titleref-memoir⟩
\NeedsTeXFormat{LaTeX2e}
\documentclass{memoir}
\usepackage{zref-titleref}[2011/03/18]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
\begin{document}
\makeatletter
\def\List{}
\def\Label#1{%
  \zref@label{#1}%
  \g@addto@macro\List{%
    \par
    #1: [\ztitleref{#1}]%
  }%
  \mbox{}%
  \zref@refused{#1}%
  \zref@ifrefundefined{#1}{%
  }{%
    \begingroup
      \edef\x{\zref@extract{#1}{title}}%
      \Expect{OK/}*{\expandafter\ltx@carthree\x{}{}{}\@nil}%
    \endgroup
  }%
}
\def\Test#1{%
  \csname#1\endcsname*{OK/#1}%
  \Label{#1*}%
  \csname#1\endcsname{OK/#1}%
  \Label{#1}%
  \csname#1\endcsname[OK/#1-toc]%
                     {WRONG-in-titleref/#1-toc-2}%
  \Label{#1-toc}%
  \expandafter\ifx\csname#1\endcsname\part
  \else
    \headnamereffalse
    \csname#1\endcsname[OK/#1-th-toc]%
                       [WRONG-in-titleref/#1-th-toc-2]%
                       {WRONG-in-titleref/#1-th-toc-3}%
    \Label{#1-th-toc}%
    \headnamereftrue
    \csname#1\endcsname[WRONG-in-titleref/#1-th-head-1]%
                       [OK/#1-th-head]%
                       {WRONG-in-titleref/#1-th-head-3}%
    \Label{#1-th-head}%
  \fi
}
\begin{qstest}{section}{section}
  \@for\x:=part,chapter,section,subsection,subsubsection\do{%
```

```
    \expandafter\Test\expandafter{\x}%
  }%
\end{qstest}
\newpage
\List
\end{document}
⟨/test-titleref-memoir⟩
```

8 Installation

8.1 Download

Package. This package is available on CTAN[2]:

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
```

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):

[2] ftp://ftp.ctan.org/tex-archive/

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

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

8.4 Refresh file name databases

If your TeX distribution (teTeX, mikTeX, ...) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlsr`.

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 LaTeX. The `.dtx` chooses its action depending on the format:

plain TeX: Run docstrip and extract the files.

LaTeX: Generate the documentation.

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

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

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

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

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

An example follows how to generate the documentation with pdfLaTeX:

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

9 References

[1] Package footmisc, Robin Fairbairns, 2004/01/23 v5.3a.CTAN:macros/latex/contrib/footmisc/footmisc.dtx

[2] Package hyperref, Sebastian Rahtz, Heiko Oberdiek, 2006/08/16 v6.75c.CTAN:macros/latex/contrib/hyperref/

[3] Package lastpage, Jeff Goldberg, 1994/06/25 v0.1b.CTAN:macros/latex/contrib/lastpage/

[4] Package nameref, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12 v2.24.CTAN:macros/latex/contrib/hyperref/nameref.dtx

[5] Package perpage, David Kastrup, 2002/12/20 v1.0.CTAN:macros/latex/contrib/bigfoot/perpage.dtx

[6] Package titleref, Donald Arsenau, 2001/04/05 v3.1.CTAN:macros/latex/contrib/misc/titleref.sty

[7] Package totpages, Wilhelm Müller, 1999/07/14 v1.00.CTAN:macros/latex/contrib/totpages/

[8] Package xr, David Carlisle, 1994/05/28 v5.02.CTAN:macros/latex/required/tools/xr.pdf

[9] Package xr-hyper, David Carlisle, 2000/03/22 v6.00beta4.CTAN:macros/latex/contrib/hyperref/xr-hyper.sty

10 History

[2006/02/20 v1.0]

- First version.

[2006/05/03 v1.1]

- Module perpage added.
- Module redesign as packages.

[2006/05/25 v1.2]

- Module dotfillmin added.
- Module base: macros `\zref@require@uniqe` 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 gettitlestring.

[2010/03/26 v2.8]

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

[2010/03/29 v2.9]

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

[2010/04/08 v2.10]

- Module xr remembers the external document name in property 'externaldocument'.

[2010/04/15 v2.11]

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

[2010/04/17 v2.12]

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

[2010/04/19 v2.13]

- `\zref@setcurrent` throws an error if the property does not exist (Florent Chervet).
- `\zref@getcurrent` the documentation is fixed (Florent Chervet). Also it returns the empty string in case of errors.
- `\zref@addprop` and `\zref@localaddprop` now take a list of property names (feature request of Florent Chervet).
- Example for `\zref@wrapper@unexpanded` corrected (Florent Chervet).

[2010/04/22 v2.14]

- Bug fix for `\zref@getcurrent` second argument wasn't eaten in case of unknown property.
- `\zref@getcurrent` supports `\zref@wrapper@unexpanded`.
- `\zref@wrapper@unexpanded` added for `\ZREF@xr@tolabel`.
- `\zref@extract`, `\zref@extractdefault`, `\zref@getcurrent` are expandable in exact two steps except inside `\zref@wrapper@unexpanded`.

[2010/04/23 v2.15]

- `\zexternaldocument` fixed for property 'url' when importing `\new@label` (bug found by Victor Ivrii).
- Two expansion steps also in `\zref@wrapper@unexpanded`.
- Nested calls of `\zref@wrapper@unexpanded` possible.

[2010/04/28 v2.16]

- More consequent use of package 'ltxcmds' and 'hologo'.
- Module pagelayout added.
- Module pageattr added.
- Robustness introduced for non-expandable interface macros.
- Internal change of the data format of property lists (suggestion of Florent Chervet).
- Module titleref: Support of environment description.

[2010/05/01 v2.17]

- `\zref@newprop` throws an error if the property already exists.
- Module xr: Bug fix for the case of several `.aux` files (bug found by Victor Ivrii).
- Module xr: Property 'urluse' and option urluse added.

[2010/05/13 v2.18]

- Module env added.
- Module savepos: `\zref@savepos` added.

[2010/10/22 v2.19]

- `\zref@addprop` and `\zref@localaddprop` are limited to one property only (incompatibility to versions v2.13 to v2.18).
- `\zref@addprops` and `\zref@localaddprops` added.
- `\zref@delprop` and `\zref@localdelprop` added.
- `\zref@labelbykv` and `\zkvlabel` (module user) with keys prop, list, delprop, immediate, values added.

[2011/02/12 v2.20]

- Fix for warning in zref-xr.

[2011/03/18 v2.21]

- Fix in module pagelayout for `\zlistpagelayout`.
- Fix for `\zref@localaddprop` (probably since v2.19).

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; plain numbers refer to the code lines where the entry is used.