

The zref package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2006/09/08 v1.3

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

6	Implementation	14
6.1	Package <code>zref</code>	14
6.1.1	Identification	14
6.1.2	Load basic module	15
6.1.3	Process options	15
6.2	Module <code>base</code>	15
6.2.1	Prefixes	15
6.2.2	Identification	16
6.2.3	Utilities	16
6.2.4	Check for ε - <code>T_EX</code>	16
6.2.5	Auxiliary file stuff	16
6.2.6	Property lists	17
6.2.7	Properties	18
6.2.8	Reference generation	19
6.2.9	Reference querying and extracting	21
6.2.10	Compatibility with <code>babel</code>	22
6.2.11	Unique counter support	23
6.2.12	Setup	23
6.3	Module <code>user</code>	24
6.4	Module <code>abspage</code>	24
6.5	Module <code>counter</code>	25
6.6	Module <code>lastpage</code>	25
6.7	Module <code>totpages</code>	25
6.8	Module <code>perpage</code>	26
6.9	Module <code>titleref</code>	28
6.9.1	Implementation	28
6.9.2	User interface	29
6.9.3	Patches for section and caption commands	30
6.10	Module <code>xr</code>	31
6.11	Module <code>hyperref</code>	35
6.12	Module <code>savepos</code>	36
6.12.1	Identification	36
6.12.2	Availability	36
6.12.3	Setup	36
6.12.4	User macros	36
6.13	Module <code>dotfill</code>	37
7	Installation	38
7.1	Some details for the interested	39
8	References	40
9	History	40
	[2006/02/20 v1.0]	40
	[2006/05/03 v1.1]	40
	[2006/05/25 v1.2]	40
	[2006/09/08 v1.3]	40
10	Index	40

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 L^AT_EX behaviour

References are created by the `\label` command:

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

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

```
\r@myref → {2.1}{7}
```

The length of the list is fixed in the L^AT_EX kernel. An interface for adding new properties is missing.

There are several tries to add new properties:

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

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

1.2 Basic idea

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

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

The entries have the following format:

```
\xref_{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 L^AT_EX3 that will need some time before its first release. Thus I have implemented it as L^AT_EX 2_ε package without disturbing the existing L^AT_EX reference system.

1.3 Interfaces

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

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

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

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

`\RequirePackage{zref-abspage}`

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

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

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

2 Interface for programmers

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

2.1 Entities

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

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

The generic format of a entry is:

`\Z@R@<refname>` \rightarrow `\<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

^{exp} means that the implementation of the marked macro is expandable.

`\zref@newlist {<listname>}`

Declares a new empty property list.

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

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

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

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

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

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

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

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

2.3 Property

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

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

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

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

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

This sets the current value of the property *<propname>*. It is a generalization of setting L^AT_EX's `\currentlabel`.

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

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

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

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

2.4 Reference generation

`\zref@label {<refname>}`

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

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

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

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

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

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

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

2.5 Data extraction

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

This is the basic command that references the value of a property `<propname>` for the reference `<refname>`. In case of errors such as undefined reference the `<default>` is used instead.

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

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

Example for page references:

```

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

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

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

`\zref@used {<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@used`, see the example file.

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

A possibility to check whether a reference exists.

`\zref@ifrefcontainspropexp {<refname>} {<propname>} {<then>} {<else>}`

Test whether a reference provides a property.

2.6 Setup

`\zref@default`

Holds the global default for unknown values.

`\zref@setdefault {<value>}`

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

`\zref@setmainlist {<value>}`

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

2.7 Declared properties

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

2.8 Wrapper for advanced situations

`\zref@wrapper@babel {...} {\langle name \rangle}`

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

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

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

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

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

```

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

```

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

2.9 Counter for unique names

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

`\zref@require@unique`

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

`\thezref@unique`

This command is used to generate unique label names.

3 User interface

3.1 Module user

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

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

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

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

`\zlabel {⟨refname⟩}^{babel}`

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

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

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

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

`\zpageref {⟨refname⟩}^{babel}`

Convenience macro, similar to `\pageref`.

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

`\zrefused {⟨refname⟩}^{babel}`

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

3.2 Module `abspage`

With the help of package `everyshi` [1] 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 option `lastpage`.

3.3 Module `lastpage`

Provides the functionality of package `lastpage` [4] in the new reference scheme. The label `LastPage` is put at the end of the document. You can refer the last page number with:

```
\zpageref{LastPage}
```

3.4 Module `totpages`

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

`\ztotpagesexp`

Prints the total number of pages or 0 if this number is not yet known. This command can also be used in calculations or counter assignments.

3.5 Module `perpage`

With `\@addtoreset` or `\numberwithin` a counter can be reset if another counter is incremented. This does not work well if the other counter is the page counter. The page counter is incremented in the output routine that is often called asynchronously 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` [6]. Also `perpage` of package `footmisc` [2] 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.

3.6 Module **counter**

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

3.7 Module **titleref**

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

`\ztitleref {⟨refname⟩}^babel`

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

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

This command allows to configure the behaviour of modul `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.

3.8 Module **savepos**

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

`\zsavepos {⟨refname⟩}`

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

$\backslash\text{zposx}^{\text{exp}} \{ \langle \text{refname} \rangle \}$ $\backslash\text{zposy}^{\text{exp}} \{ \langle \text{refname} \rangle \}$
--

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

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

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

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

3.9 Module `dotfill`

<code>\zdotfill</code>

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.

<code>\zdotfillsetup {key₁=value₁, key₂=value₂, ...}</code>

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`).

3.10 Module `xr`

This package provides the functionality of package `xr`, see [9]. It also supports the syntax of `xr-hyper`.

<code>\xexternaldocument * [<i><prefix></i>] ^{babel} {<i><external document></i>} [<i><url></i>]</code>
--

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

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

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

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

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

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

`\zref@xr@ext`

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

4 ToDo

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

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

5 Example

```

1 <example>
2 \documentclass{book}
3
4 \usepackage[ngerman]{babel}%
5
6 \usepackage[savepos,totpages,titleref,dotfill,counter,user]{zref}
7
```

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.

```

8 \makeatletter
9 \zref@newprop{chaptitle}{}
10 \zref@addprop{main}{chaptitle}
11
12 \newcommand*{\ChapterStart}[2]{%
13   \cleardoublepage
14   \def\current@chapid{#1}%
15   \zref@setcurrent{chaptitle}{#2}%
16   \chapter{#2}%
17   \zlabel{chap:#1}%
18 }
19 \newcommand*{\ChapterStop}{%
20   \cleardoublepage
```

```

21 \zref@wrapper@immediate{%
22   \zref@labelbyprops{chapend:\current@chapid}{abspage}%
23 }%
24 }

```

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

```

25 \newcommand*{\ChapterPages}[1]{%
26   \zrefused{chap:#1}%
27   \zrefused{chapend:#1}%
28   \number\numexpr
29     \zref@extract{chapend:#1}{abspage}%
30     -\zref@extract{chap:#1}{abspage}%
31     +1\relax
32 }
33 \makeatother
34 \begin{document}

```

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

```

35 \makeatletter
36
37 \frontmatter
38 \zlabel{documentstart}
39
40 \begin{itemize}
41 \item
42   The frontmatter part has
43   \number\numexpr\zref@extract{chap:first}{abspage}-1\relax~pages.
44 \item
45   Chapter \zref{chap:first} has \ChapterPages{first} page(s).
46 \item
47   Section \zref{hello} is on the
48   \ifcase\numexpr
49     \zref@extractdefault{hello}{page}{0}%
50     -\zref@extractdefault{chap:first}{page}{0}%
51     +1\relax
52     ??\or first\or second\or third\or forth\fi
53   ~page inside its chapter.
54 \item
55   The document has
56   \zref[abspage]{LastPage} pages.
57   This number is \ifodd\ztotpages odd\else even\fi.
58 \item
59   The last page is labeled with \zpageref{LastPage}.
60 \item
61   The title of chapter \zref{chap:next} is ‘‘\zref[chaptitle]{chap:next}’’.
62 \end{itemize}
63
64 \tableofcontents
65
66 \mainmatter
67 \ChapterStart{first}{First chapter}
68

```

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`.

```

69 \section{Test}
70 \zlabel{a"o}
71 Section \zref{a"o} on page
72 \zref@wrapper@babel\zref@extract{a"o}{page}.
73
74 Text.

```

```

75 \newpage
76
77 \section{Hello World}
78 \zlabel{hello}
79
80 \ChapterStop
81
82 \ChapterStart{next}{Next chapter with \emph{umlauts}: "a"o"u"s}
83

```

Here an example follows that makes use of pdf_TE_X's “savepos” feature. The position on the page is not known before the page is constructed and shipped out. Therefore the position is stored in references and are available for calculations in the next L^AT_EX compile run.

```

84 The width of the first column is
85 \the\dimexpr \zposx{secondcol}sp - \zposx{firstcol}sp\relax,\,
86 the height difference of the two baselines is
87 \the\dimexpr \zposy{firstcol}sp - \zposy{secondline}sp\relax:\,
88 \begin{tabular}{ll}
89 \zsavapos{firstcol}Hello&\zsavapos{secondcol}World\,
90 \zsavapos{secondline}Second line&foobar\,
91 \end{tabular}
92

```

With \zrefused L^AT_EX is notified, if the references are not yet available and L^AT_EX can generate the rerun hint.

```

93 \zrefused{firstcol}
94 \zrefused{secondcol}
95 \zrefused{secondline}
96
97 \ChapterStop
Test for module \dotfill.
98 \ChapterStart{dotfill}{Test for dotfill feature}
99 \newcommand*{\dfptest}[1]{%
100   #1&
101   [\makebox[#1]{\dotfill}]&
102   [\makebox[#1]{\zdotfill}]\,
103 }
104 \begin{tabular}{rll}
105 & [\verb|\dotfill|] & [\verb|\zdotfill|]\,
106 \dfptest{0.43em}
107 \dfptest{0.44em}
108 \dfptest{0.45em}
109 \dfptest{0.87em}
110 \dfptest{0.88em}
111 \dfptest{0.89em}
112 \dfptest{1.31em}
113 \dfptest{1.32em}
114 \dfptest{1.33em}
115 \end{tabular}
116 \ChapterStop
117 \end{document}
118 \</example>

```

6 Implementation

6.1 Package zref

6.1.1 Identification

```

119 \<*package>
120 \NeedsTeXFormat{LaTeX2e}
121 \ProvidesPackage{zref}

```

6.1.2 Load basic module

```
123 \RequirePackage{zref-base}[2006/09/08]
```

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

```
124 \ifundefined{ZREF@baseok}{\endinput}{}
```

6.1.3 Process options

Known modules are loaded and the release date is checked.

```
125 \def\ZREF@temp#1{%
126   \DeclareOption{#1}{%
127     \AtEndOfPackage{%
128       \RequirePackage{zref-#1}[2006/09/08]%
129     }%
130   }%
131 }
132 \ZREF@temp{abspage}
133 \ZREF@temp{counter}
134 \ZREF@temp{dotfill}
135 \ZREF@temp{hyperref}
136 \ZREF@temp{lastpage}
137 \ZREF@temp{perpage}
138 \ZREF@temp{savepos}
139 \ZREF@temp{titleref}
140 \ZREF@temp{totpages}
141 \ZREF@temp{user}
142 \ZREF@temp{xr}

143 \ProcessOptions\relax
144 \endpackage
```

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 option *user* is set.

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

ZREF@TR: Setup for titleref.

6.2.2 Identification

```
145 <*base>
146 \NeedsTeXFormat{LaTeX2e}
147 \ProvidesPackage{zref-base}%
148 [2006/09/08 Module base for zref (H0)]
```

6.2.3 Utilities

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

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

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

```
150 \def\ZREF@ErrorNoLine#1#2{%
151   \begingroup
152     \let\on@line\@empty
153     \PackageError\ZREF@name{#1}{#2}%
154   \endgroup
155 }
```

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

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

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

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

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

```
158 \def\ZREF@patch#1{%
159   \begingroup\expandafter\expandafter\expandafter\endgroup
160   \expandafter\ifx\csname #1\endcsname\relax
161     \expandafter\@gobble
162   \else
163     \expandafter\let\csname ZREF@org@#1\endcsname
164     \csname #1\endcsname
165     \expandafter\@firstofone
166   \fi
167 }
```

6.2.4 Check for ε -TeX

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

```
168 \begingroup
169 \@ifundefined{eTeXversion}{%
170   \ZREF@ErrorNoLine{%
171     Missing support for eTeX; package is abandoned%
172   }{%
173     Use a TeX compiler that support eTeX and enable eTeX %
174     in the format.%
175   }%
176 \endgroup
177 \endinput
178 }{}
179 \endgroup
```

6.2.5 Auxiliary file stuff

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

```
180 \RequirePackage{auxhook}
```

```

181 \AddLineBeginAux{%
182   \string\providecommand\string\zref@newlabel[2]{}%
183 }

```

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

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

```

184 \def\zref@newlabel{%
185   \@newl@bel{Z@R}%
186 }

```

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.

```

187 \def\zref@newlist#1{%
188   \zref@iflistundefined{#1}{%
189     \ifdefinable{Z@L@#1}{%
190       \global\expandafter\let\csname Z@L@#1\endcsname\@empty
191       \PackageInfo{zref}{New property list: #1}%
192     }%
193   }{%
194     \PackageError{ZREF@name}{
195       Property list ‘#1’ already exists%
196     }\@ehc
197   }%
198 }

```

`\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.

```

199 \def\zref@iflistundefined#1{%
200   \expandafter\ifx\csname Z@L@#1\endcsname\relax
201     \expandafter\@firstoftwo
202   \else
203     \expandafter\@secondoftwo
204   \fi
205 }

```

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

```

206 \def\zref@listexists#1{%
207   \zref@iflistundefined{#1}{%
208     \PackageError{ZREF@name}{
209       Property list ‘#1’ does not exist%
210     }\@ehc
211   }%
212 }

```

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

```

213 \def\zref@listcontainsprop#1{%
214   \expandafter\ZREF@listcontainsprop\csname Z@L@#1\endcsname
215 }
216 \def\ZREF@listcontainsprop#1#2{%
217   \begingroup

```

```

218 \ZREF@foundfalse
219 \edef\y{#2}%
220 \@tfor\x:=#1\do{%
221 \edef\x{\x}%
222 \ifx\x\y
223 \ZREF@foundtrue
224 \fi
225 }%
226 \expandafter\endgroup
227 \ifZREF@found
228 \expandafter\@firstoftwo
229 \else
230 \expandafter\@secondoftwo
231 \fi
232 }

```

\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.

```

233 \def\zref@addprop#1#2{%
234 \zref@listexists{#1}{%
235 \zref@propexists{#2}{%
236 \zref@listcontainsprop{#1}{#2}{%
237 \PackageWarning\ZREF@name{%
238 Property ‘#2’ is already in list ‘#1’%
239 }%
240 }{%
241 \expandafter\g@addto@macro\csname Z@L@#1\endcsname{{#2}}%
242 }%
243 }%
244 }%
245 }

```

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.

```

246 \def\zref@ifpropundefined#1{%
247 \expandafter\ifx\csname Z@E@#1\endcsname\relax
248 \expandafter\@firstoftwo
249 \else
250 \expandafter\@secondoftwo
251 \fi
252 }

```

\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.

```

253 \def\zref@propexists#1{%
254 \zref@ifpropundefined{#1}{%
255 \PackageError\ZREF@name{%
256 Property ‘#1’ does not exist%
257 }\@ehc
258 }%
259 }

```

\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.

```

260 \def\zref@newprop{%
261   \@ifstar{%
262     \let\ZREF@X\noexpand
263     \ZREF@newprop
264   }{%
265     \let\ZREF@X\empty
266     \ZREF@newprop
267   }%
268 }
269 \def\ZREF@newprop#1{%
270   \PackageInfo{zref}{New property: #1}%
271   \def\ZREF@P{#1}%
272   \ifnextchar[\ZREF@@newprop{\ZREF@@newprop[\zref@default]}%
273 }
274 \def\ZREF@@newprop[#1]{%
275   \global\@namedef{Z@D@\ZREF@P}{#1}%
276   \global\expandafter\let\csname Z@X@\ZREF@P\endcsname\ZREF@X
277   \expandafter\ZREF@@@newprop\csname\ZREF@P\endcsname
278   \zref@setcurrent\ZREF@P
279 }
280 \def\ZREF@@@newprop#1{%
281   \expandafter\gdef\csname Z@E@\ZREF@P\endcsname##1#1##2##3\ZREF@nil{##2}%
282 }

```

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

```

283 \def\zref@setcurrent#1{%
284   \expandafter\def\csname Z@C@#1\endcsname
285 }

```

6.2.8 Reference generation

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

```

286 \def\zref@label#1{%
287   \zref@labelbylist{#1}\ZREF@mainlist
288 }

```

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

```

289 \def\zref@labelbylist#1#2{%
290   \@bsphack
291   \zref@listexists{#2}{%
292     \expandafter\expandafter\expandafter\ZREF@label
293     \expandafter\expandafter\expandafter{%
294       \csname Z@L@#2\endcsname
295     }{#1}%
296   }%
297   \@esphack
298 }

```

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

```

299 \def\zref@labelbyprops#1#2{%
300   \@bsphack
301   \begingroup
302     \edef\l{#2}%
303     \toks@{}%
304     \@for\x:=#2\do{%
305       \zref@ifpropundefined{x}{%
306         \PackageWarning\ZREF@name{%

```

```

307         Property ‘\x’ is not known%
308     }%
309 }{%
310     \toks@\expandafter\expandafter\expandafter{%
311         \expandafter\the\expandafter\toks@\expandafter{\x}%
312     }%
313 }%
314 }%
315 \expandafter\endgroup
316 \expandafter\ZREF@label\expandafter{\the\toks@}{#1}%
317 \@esphack
318 }

```

`\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.

```

319 \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`.

```

320 \long\def\zref@wrapper@immediate#1{%
321     \begingroup
322     \ZREF@immediatetrue
323     \let\ZREF@org@write\write
324     \def\write{\immediate\ZREF@org@write}%
325     #1%
326 \endgroup
327 }

```

`\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 20 is made expandable in this case.

```

328 \def\ZREF@label#1#2{%
329     \if@filesw
330         \begingroup
331         \ifZREF@immediate
332             \let\ZREF@org@thepage\thepage
333         \fi
334         \protected@write\@auxout{%
335             \ifZREF@immediate
336                 \let\thepage\ZREF@org@thepage
337             \fi
338             \let\ZREF@temp\@empty
339             \@tfor\ZREF@P:=#1\do{%
340                 \expandafter\ifx
341                     \csname\ifZREF@immediate relax\else Z@X@\ZREF@P\fi\endcsname
342                 \noexpand
343                 \expandafter\let\csname Z@C@\ZREF@P\endcsname\relax
344             \fi
345             \toks@\expandafter{\ZREF@temp}%
346             \edef\ZREF@temp{%
347                 \the\toks@
348                 \expandafter\string\csname\ZREF@P\endcsname{%
349                     \expandafter\noexpand\csname Z@C@\ZREF@P\endcsname
350                 }%
351             }%
352         }%
353     }{%
354         \string\zref@newlabel{#2}{\ZREF@temp}%
355     }%

```

```

356 \endgroup
357 \fi
358 }
359 \def\ZREF@addtoks#1{%
360 \toks@\expandafter\expandafter\expandafter{%
361 \expandafter\the\expandafter\toks@#1%
362 }%
363 }

```

6.2.9 Reference querying and extracting

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

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

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

```

364 \def\zref@ifrefundefined#1{%
365 \expandafter\ifx\csname Z@R@#1\endcsname\relax
366 \expandafter\@firstoftwo
367 \else
368 \expandafter\@secondoftwo
369 \fi
370 }

```

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

```

371 \def\zref@refused#1{%
372 \begingroup
373 \csname @safe@activetrue\endcsname
374 \zref@ifrefundefined{#1}{%
375 \protect\G@refundefinedtrue
376 \@latex@warning{%
377 Reference ‘#1’ on page \thepage \space undefined%
378 }%
379 }{}%
380 \endgroup
381 }

```

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

```

382 \def\zref@extract#1#2{%
383 \expandafter\expandafter\expandafter\ZREF@extract
384 \expandafter\expandafter\expandafter{%
385 \csname Z@D@#2\endcsname
386 }{#1}{#2}%
387 }
388 \def\ZREF@extract#1#2#3{%
389 \zref@extractdefault{#2}{#3}{#1}%
390 }

```

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

```

391 \def\zref@ifrefcontainsprop#1#2{%
392   \zref@ifrefundefined{#1}{%
393     \@secondoftwo
394   }{%
395     \expandafter\ZREF@ifrefcontainsprop
396     \csname Z@E@#2\expandafter\endcsname
397     \csname#2\expandafter\expandafter\expandafter\endcsname
398     \expandafter\expandafter\expandafter{%
399       \csname Z@R@#1\endcsname
400     }%
401   }%
402 }
403 \def\ZREF@ifrefcontainsprop#1#2#3{%
404   \expandafter\ifx\expandafter\ZREF@novalue
405   #1#3#2\ZREF@novalue\ZREF@nil\@empty
406   \expandafter\@secondoftwo
407   \else
408   \expandafter\@firstoftwo
409   \fi
410 }
411 \def\ZREF@novalue{\ZREF@NOVALUE}

```

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

```

412 \def\zref@extractdefault#1#2#3{%
413   \zref@ifrefundefined{#1}{%
414     \ZREF@unexpanded{#3}%
415   }{%
416     \expandafter\expandafter\expandafter\ZREF@unexpanded
417     \expandafter\expandafter\expandafter{%
418       \csname Z@E@#2\expandafter\expandafter\expandafter\endcsname
419       \csname Z@R@#1\expandafter\expandafter\expandafter\endcsname
420       \csname#2\endcsname{#3}\ZREF@nil
421     }%
422   }%
423 }

```

`\zref@wrapper@unexpanded`

```

424 \long\def\zref@wrapper@unexpanded#1{%
425   \let\ZREF@unexpanded\unexpanded
426   #1%
427   \let\ZREF@unexpanded\@firstofone
428 }
429 \let\ZREF@unexpanded\@firstofone

```

6.2.10 Compatibility with babel

`\zref@wrapper@babel`

```

430 \long\def\zref@wrapper@babel#1#2{%
431   \begingroup
432     \csname @safe@activetrue\endcsname
433     \edef\x{#2}%
434   \expandafter\endgroup
435   \expandafter\ZREF@wrapper@babel\expandafter{\x}{#1}%
436 }
437 \def\ZREF@wrapper@babel#1#2{%
438   #2{#1}%
439 }

```

6.2.11 Unique counter support

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

```

440 \def\zref@require@unique{%
441   \ifundefined{c@zref@unique}{%
442     \newcounter{zref@unique}%

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

443   \renewcommand*{\thezref@unique}{%
444     zref@number\c@zref@unique
445   }%
446 }{}%
447 }
```

6.2.12 Setup

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

```

448 \def\zref@setdefault#1{%
449   \def\zref@default{#1}%
450 }

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

451 \zref@setdefault{%
452   \nfss@text{\reset@font\bfseries ??}%
453 }
```

Main property list.

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

```

454 \def\zref@setmainlist#1{%
455   \def\ZREF@mainlist{#1}%
456 }
457 \zref@setmainlist{main}

Now we create the list.

458 \zref@newlist\ZREF@mainlist
```

Main properties. The two properties `default` and `page` are created and added to the main property list. They store the data that standard L^AT_EX uses in its references created by `\label`.

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

`page` the apperance of the page counter

```

459 \zref@newprop{default}{\@currentlabel}
460 \zref@newprop*{page}{\thepage}
461 \zref@addprop\ZREF@mainlist{default}
462 \zref@addprop\ZREF@mainlist{page}
```

Mark successful loading

```

463 \let\ZREF@baseok\@empty
464 \</base>
```

6.3 Module user

```

465 <*user>
466 \NeedsTeXFormat{LaTeX2e}
467 \ProvidesPackage{zref-user}%
468 [2006/09/08 v1.3 Module user for zref (HO)]
469 \RequirePackage{zref-base}[2006/09/08]
470 \@ifundefined{ZREF@baseok}{\endinput}{}

```

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

First we define the pendants to the standard L^AT_EX referencing commands `\label`, `\ref`, and `\pageref`.

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

```

471 \newcommand*\zlabel{%
472   \ifx\label\@gobble
473     \expandafter\@gobble
474   \else
475     \expandafter\zref@wrapper@babel\expandafter\zref@label
476   \fi
477 }%

```

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

```

478 \newcommand*\zref}[2][default]{%
479   \zref@propexists{#1}{%
480     \zref@wrapper@babel\ZREF@zref{#2}{#1}%
481   }%
482 }%
483 \def\ZREF@zref#1{%
484   \zref@refused{#1}%
485   \zref@extract{#1}%
486 }%

```

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

```

487 \newcommand*\zpageref{%
488   \zref[page]%
489 }%

```

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

```

490 \newcommand*\zrefused{\zref@refused}%
491 </user>

```

6.4 Module abspage

```

492 <*abspage>
493 \NeedsTeXFormat{LaTeX2e}
494 \ProvidesPackage{zref-abspage}%
495 [2006/09/08 v1.3 Module abspage for zref (HO)]
496 \RequirePackage{zref-base}[2006/09/08]
497 \@ifundefined{ZREF@baseok}{\endinput}{}

```

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

```

498 \RequirePackage{everyshi}%
499 \newcounter{abspage}%
500 \setcounter{abspage}{0}%
501 \EveryShipout{%

```

```

502 \stepcounter{abspage}%
503 }%
504 \zref@newprop*{abspage}[0]{\the\c@abspage}%
505 \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.

```

506 \end{abspage}

```

6.5 Module counter

```

507 \newcommand*{counter}
508 \NeedsTeXFormat{LaTeX2e}
509 \ProvidesPackage{zref-counter}%
510 [2006/09/08 v1.3 Module counter for zref (HO)]
511 \RequirePackage{zref-base}[2006/09/08]
512 \ifundefined{ZREF@baseok}{\endinput}{}

```

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

```

513 \zref@newprop{counter}{}
514 \zref@addprop\ZREF@mainlist{counter}

```

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

```

515 \AtBeginDocument{%
516 \ZREF@patch{refstepcounter}{%
517 \def\refstepcounter#1{%
518 \zref@setcurrent{counter}{#1}%
519 \ZREF@org@refstepcounter{#1}%
520 }%
521 }%
522 }
523 \end{counter}

```

6.6 Module lastpage

```

524 \newcommand*{lastpage}
525 \NeedsTeXFormat{LaTeX2e}
526 \ProvidesPackage{zref-lastpage}%
527 [2006/09/08 v1.3 Module lastpage for zref (HO)]
528 \RequirePackage{zref-base}[2006/09/08]
529 \ifundefined{ZREF@baseok}{\endinput}{}

```

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

```

530 \AtBeginDocument{%
531 \AtEndDocument{%
532 \if@filesw
533 \clearpage
534 \begingroup
535 \advance\c@page\m@ne
536 \zref@wrapper@immediate{\zref@label{LastPage}}%
537 \endgroup
538 \fi
539 }%
540 }
541 \end{lastpage}

```

6.7 Module totpages

```

542 \newcommand*{totpages}

```

```

543 \NeedsTeXFormat{LaTeX2e}
544 \ProvidesPackage{zref-totpages}%
545 [2006/09/08 v1.3 Module totpages for zref (HO)]
546 \RequirePackage{zref-base}[2006/09/08]
547 \@ifundefined{ZREF@baseok}{\endinput}{\}

```

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

```

548 \RequirePackage{zref-abspage}[2006/09/08]
549 \RequirePackage{zref-lastpage}[2006/09/08]

```

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

```

550 \newcommand*{\ztotpages}{%
551   \zref@extractdefault{LastPage}{abspage}{0}%
552 }

```

Also we mark the reference `LastPage` as used:

```

553 \AtBeginDocument{%
554   \zref@refused{LastPage}%
555 }
556 </totpages>

```

6.8 Module `perpage`

```

557 <*perpage>
558 \NeedsTeXFormat{LaTeX2e}
559 \ProvidesPackage{zref-perpage}%
560 [2006/09/08 v1.3 Module perpage for zref (HO)]
561 \RequirePackage{zref-base}[2006/09/08]
562 \@ifundefined{ZREF@baseok}{\endinput}{\}

```

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

For detecting changed pages module `abspage` is loaded.

```

563 \RequirePackage{zref-abspage}[2006/09/08]

```

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

```

564 \zref@newprop*{pagevalue}[0]{\number\c@page}
565 \zref@newlist{perpage}
566 \zref@addprop{perpage}{abspage}
567 \zref@addprop{perpage}{page}
568 \zref@addprop{perpage}{pagevalue}

```

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

```

569 \newcounter{zpage}

```

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

```

570 \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`.

```

571 \let\ZREF@org@stepcounter\stepcounter
572 \def\stepcounter#1{%
573   \ifcsname @stepcounterhook@#1\endcsname
574     \csname @stepcounterhook@#1\endcsname
575   \fi
576   \ZREF@org@stepcounter{#1}%
577 }

```

`\zmakeperpage` Makro `\zmakeperpage` resets a counter at each page break. It uses the same syntax and semantics as `\MakePerPage` from package `perpage` [6]. 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.

```
578 \newcommand*{\zmakeperpage}{%
579   \@ifnextchar[\ZREF@makeperpage@opt{\ZREF@@makeperpage[\z@]}%
580 }
```

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

```
581 \def\ZREF@makeperpage@opt[#1]{%
582   \begingroup
583     \edef\x{\endgroup
584       \noexpand\ZREF@@makeperpage[\number\numexpr#1-1\relax]%
585     }%
586   \x
587 }

588 \def\ZREF@@makeperpage[#1]#2{%
589   \@ifundefined{@stepcounterhook@#2}{%
590     \expandafter\gdef\csname @stepcounterhook@#2\endcsname{%
591     }{%}%
592     \expandafter\gdef\csname ZREF@perpage@#2\endcsname{%
593       \ZREF@perpage@step{#2}{#1}%
594     }%
595     \expandafter\g@addto@macro\csname @stepcounterhook@#2\endcsname{%
596       \ifcsname ZREF@perpage@#2\endcsname
597         \csname ZREF@perpage@#2\endcsname
598       \fi
599     }%
600 }
```

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

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

First the reference is generated.

```
602   \global\advance\c@zref@unique\@ne
603   \begingroup
604     \expandafter\zref@labelbylist\expandafter{\thezref@unique}{perpage}%
```

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

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

```
605     \zref@ifrefundefined\thezref@unique{%
606       \global\c@zpage=\c@page
607       \global\let\thezpage\thepage
608       \expandafter\xdef\csname ZREF@abspage@#1\endcsname{\number\c@abspage}%
609     }{%
```

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

```
610       \global\c@zpage=\zref@extract\thezref@unique{pagevalue}\relax
611       \xdef\thezpage{\noexpand\zref@extract{\thezref@unique}{page}}%
612       \expandafter\xdef\csname ZREF@abspage@#1\endcsname{%
613         \zref@extractdefault\thezref@unique{abspage}{\number\c@abspage}%
614       }%
615     }%
```

Page changes are detected by a changed absolute page number.

```
616     \expandafter\ifx\csname ZREF@abspage@#1\endcsname
617       \csname ZREF@currentabspage@#1\endcsname
618   \else
619     \global\csname c@#1\endcsname=#2\relax
620     \global\expandafter\let
621       \csname ZREF@currentabspage@#1\endcsname
622       \csname ZREF@abspage@#1\endcsname
```

```

623 \fi
624 \endgroup
625 }

```

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

```

626 \newcommand*{\zunmakeperpage}[1]{%
627 \global\expandafter\let\csname ZREF@perpage@#1\endcsname\@undefined
628 }

629 \</perpage>

```

6.9 Module `titleref`

```

630 \*titleref
631 \NeedsTeXFormat{LaTeX2e}
632 \ProvidesPackage{zref-titleref}%
633 [2006/09/08 v1.3 Module titleref for zref (H0)]
634 \RequirePackage{zref-base}[2006/09/08]
635 \@ifundefined{ZREF@baseok}{\endinput}{}

```

6.9.1 Implementation

```

636 \RequirePackage{keyval}

```

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

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

```

637 \let\zref@titleref@current\@empty

```

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

```

638 \zref@newprop{title}{\zref@titleref@current}%
639 \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.

```

640 \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`.

```

641 \let\ZREF@titleref@hook\@empty

```

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

```

642 \def\zref@titleref@cleanup#1{%
643 \begingroup
644 \toks@\expandafter{%
645 \ZREF@titleref@hook
646 #1%
647 }%
648 \expandafter\endgroup

```

```

649 \expandafter\def\expandafter\ZREF@titleref@hook\expandafter{%
650   \the\toks@
651 }%
652 }%

```

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

```

653 \newif\ifzref@titleref@stripperperiod
654 \zref@titleref@stripperperiodtrue

```

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

```

655 \def\zref@titleref@setcurrent#1{%
656   \def\zref@titleref@current{#1}%
657   \ifzref@titleref@expand
658     \begingroup
659     \let\label\@gobble
660     \let\index\@gobble
661     \let\glossary\@gobble
662     \let\markboth\@gobbletwo
663     \let\@mkboth\@gobbletwo
664     \let\markright\@gobble
665     \let\protect\@unexpandable@protect
666     \ZREF@titleref@hook
667     \edef\x{\endgroup
668       \noexpand\def\noexpand\zref@titleref@current{%
669         \zref@titleref@current
670       }%
671     }%
672     \x
673   \fi
674   \edef\zref@titleref@current{%
675     \detokenize\expandafter{\zref@titleref@current}%
676   }%
677   \ifzref@titleref@stripperperiod
678     \edef\zref@titleref@current{%
679       \expandafter\ZREF@stripperperiod\zref@titleref@current
680       \@empty.\@empty\@nil
681     }%
682   \fi
683 }%

```

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

```

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

```

6.9.2 User interface

`\ztitlerefsetup` The behaviour of option `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`.

```

685 \define@key{ZREF@TR}{expand}[true]{%
686   \csname zref@titleref@expand#1\endcsname
687 }%
688 \define@key{ZREF@TR}{stripperperiod}[true]{%
689   \csname zref@titleref@stripperperiod#1\endcsname
690 }%
691 \define@key{ZREF@TR}{cleanup}{%
692   \zref@titleref@cleanup{#1}%

```

```

693 }%
694 \define@key{ZREF@TR}{title}{%
695   \def\zref@titleref@current{#1}%
696 }%
697 \newcommand*\ztitlerefsetup{%
698   \setkeys{ZREF@TR}%
699 }%

```

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

```

700 \newcommand*\ztitleref{%
701   \zref@wrapper@babel\ZREF@titleref
702 }%
703 \def\ZREF@titleref#1{%
704   \begingroup
705     \zref@refused{#1}%
706     \let\label\@gobble
707     \zref@extract{#1}{title}%
708   \endgroup
709 }%

```

6.9.3 Patches for section and caption commands

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

Captions of figures and tables.

```

710 \AtBeginDocument{%
711   \ZREF@patch{@caption}{%
712     \long\def\@caption#1[#2]{%
713       \zref@titleref@setcurrent{#2}%
714       \ZREF@org@@caption{#1}[{#2}]%
715     }%
716   }%

```

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

```

717   \ZREF@patch{@part}{%
718     \def\@part[#1]{%
719       \zref@titleref@setcurrent{#1}%
720       \ZREF@org@@part[{#1}]%
721     }%
722   }%
723   \ZREF@patch{@chapter}{%
724     \def\@chapter[#1]{%
725       \zref@titleref@setcurrent{#1}%
726       \ZREF@org@@chapter[{#1}]%
727     }%
728   }%
729   \ZREF@patch{@sect}{%
730     \def\@sect#1#2#3#4#5#6[#7]{%
731       \zref@titleref@setcurrent{#7}%
732       \ZREF@org@@sect{#1}{#2}{#3}{#4}{#5}{#6}[{#7}]%
733     }%
734   }%

```

The star versions of the section commands.

```

735   \ZREF@patch{@spart}{%
736     \def\@spart#1{%
737       \zref@titleref@setcurrent{#1}%
738       \ZREF@org@@spart{#1}%
739     }%
740   }%
741   \ZREF@patch{@schapter}{%
742     \def\@schapter#1{%

```

```

743     \zref@titleref@setcurrent{#1}%
744     \ZREF@org@@schapter{#1}%
745 }%
746 }%
747 \ZREF@patch{@ssect}{%
748   \def\ssect#1#2#3#4#5{%
749     \zref@titleref@setcurrent{#5}%
750     \ZREF@org@@ssect{#1}{#2}{#3}{#4}{#5}%
751   }%
752 }%
Package titlesec.
753 \@ifpackageloaded{titlesec}{%
754   \ZREF@patch{ttl@sect@i}{%
755     \def\ttl@sect@i#1#2[#3]#4{%
756       \zref@titlesec@setcurrent{#4}%
757       \ZREF@org@ttl@sect@i{#1}{#2}[#3]{#4}%
758     }%
759   }%
760 }{}%
761 }%
762 </titleref>

```

6.10 Module xr

```

763 <*xr>
764 \NeedsTeXFormat{LaTeX2e}
765 \ProvidesPackage{zref-xr}%
766 [2006/09/08 v1.3 Module xr for zref (HO)]
767 \RequirePackage{zref-base}[2006/09/08]
768 \@ifundefined{ZREF@baseok}{\endinput}{}
769 \RequirePackage{keyval}

```

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

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

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

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

```

771 \providecommand*{\zref@xr@ext}{}%
772 \@ifundefined{XR@ext}{pdf}{\XR@ext}%
773 }%

```

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

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

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

```

775 \newcommand*{\xexternaldocument}{%
776   \begingroup
777   \csname @safe@actives@true\endcsname
778   \makeatletter
779   \@ifstar{%
780     \ZREF@xr@zreflabelfalse
781     \@testopt\ZREF@xr@externaldocument{}%
782   }{%
783     \ZREF@xr@zreflabeltrue
784     \@testopt\ZREF@xr@externaldocument{}%

```

```

785 }%
786 }%

```

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

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

```

787 \def\ZREF@xr@externaldocument[#1]#2{%
788   \def\ZREF@xr@prefix{#1}%
789   \let\ZREF@xr@filelist\@empty
790   \edef\ZREF@xr@file{#2.aux}%
791   \filename@parse{#2}%
792   \@testopt\ZREF@xr@graburl{#2.\zref@xr@ext}%
793 }%
794 \def\ZREF@xr@graburl[#1]{%
795   \edef\ZREF@xr@url{#1}%
796   \ZREF@xr@checkfile
797   \endgroup
798 }%

```

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

```

799 \def\ZREF@xr@checkfile{%
800   \openin\@inputcheck\ZREF@xr@file\relax
801   \ifeof\@inputcheck
802     \PackageWarning{zref/xr}{%
803       File '\ZREF@xr@file' not found or empty,\MessageBreak
804       labels not imported%
805     }%
806   \else
807     \PackageInfo{zref/xr}{%
808       Label \ifZREF@xr@zreflabel (zref) \fi import from '\ZREF@xr@file'%
809     }%
810     \def\ZREF@xr@found{0}%
811     \def\ZREF@xr@ignored{0}%
812     \ZREF@xr@processfile
813     \closein\@inputcheck
814     \begingroup
815       \let\on@line\@empty
816       \PackageInfo{zref/xr}{%
817         Statistics for '\ZREF@xr@file':
818         \ZREF@xr@found\space found, %
819         \ZREF@xr@ignored\space ignored%
820       }%
821     \endgroup
822   \fi
823   \ifx\ZREF@xr@filelist\@empty
824     \else
825       \edef\ZREF@xr@file{\expandafter\@car\ZREF@xr@filelist\@nil}%
826       \edef\ZREF@xr@filelist{\expandafter\@cdr\ZREF@xr@filelist\@nil}%
827       \expandafter\ZREF@xr@checkfile
828     \fi
829 }%

```

`\ZREF@xr@processfile`

```

830 \def\ZREF@xr@processfile{%
831   \read\@inputcheck to\ZREF@xr@line
832   \expandafter\ZREF@xr@processline\ZREF@xr@line..\ZREF@nil
833   \ifeof\@inputcheck

```

```

834 \else
835 \expandafter\ZREF@xr@procesfile
836 \fi
837 }%

\ZREF@xr@processline The most work must be done for analyzing the arguments of \newlabel.
838 \long\def\ZREF@xr@processline#1#2#3\ZREF@nil{%
839 \def\x{#1}%
840 \toks@{#2}%
841 \ifZREF@xr@zreflabel
842 \ifx\x\ZREF@xr@zref@newlabel
843 \expandafter\ZREF@xr@process@zreflabel\ZREF@xr@line...\ZREF@nil
844 \fi
845 \else
846 \ifx\x\ZREF@xr@newlabel
847 \expandafter\ZREF@xr@process@label\ZREF@xr@line...[]\ZREF@nil
848 \fi
849 \fi
850 \ifx\x\ZREF@xr@@input
851 \edef\ZREF@xr@filelist{%
852 \unexpanded\expandafter{\ZREF@xr@filelist}%
853 {\filename@area\the\toks@}%
854 }%
855 \fi
856 \ifeof\@inputcheck
857 \else
858 \expandafter\ZREF@xr@procesfile
859 \fi
860 }%
861 \def\ZREF@xr@process@zreflabel\zref@newlabel#1#2#3\ZREF@nil{%
862 \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
863 \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
864 \def\x{#2}%
865 \@ifundefined{\ZREF@xr@refname}{%
866 \let\ZREF@xr@list\x
867 \ifx\ZREF@xr@list\@empty
868 \PackageWarningNoLine{zref/xr}{%
869 Label ‘#1’ without properties ignored\MessageBreak
870 in file ‘\ZREF@xr@file’%
871 }%
872 \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
873 \else
874 \expandafter\ZREF@xr@checklist\x\ZREF@nil
875 \expandafter\global\expandafter\let
876 \csname \ZREF@xr@refname\endcsname\x
877 \fi
878 \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
879 }{%
880 \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
881 }%
882 }%
883 \def\ZREF@xr@process@label\newlabel#1#2#3[#4]#5\ZREF@nil{%
884 \def\ZREF@xr@refname{Z@R@\ZREF@xr@prefix#1}%
885 \edef\ZREF@xr@found{\the\numexpr\ZREF@xr@found+1\relax}%
886 \def\x{#2}%
887 \@ifundefined{\ZREF@xr@refname}{%
888 \expandafter\ZREF@xr@scanparams
889 \csname\ZREF@xr@refname\expandafter\endcsname
890 \x{}{}{}{}{}\ZREF@nil
891 \ifx\\#4\\%
892 \else
893 % ntheorem knows an optional argument at the end of \newlabel
894 \zref@ifpropundefined{theotype}{%

```

```

895      \zref@newprop{theotype}{}%
896    }{}%
897    \expandafter\g@addto@macro
898      \csname\ZREF@xr@refname\endcsname{\theotype{#4}}%
899    \fi
900    \ZREF@xr@urlcheck{\ZREF@xr@prefix#1}%
901  }{}%
902    \ZREF@xr@ignorewarning{\ZREF@xr@prefix#1}%
903  }%
904 }
905 \def\ZREF@xr@zref@newlabel{\zref@newlabel}%
906 \def\ZREF@xr@newlabel{\newlabel}%
907 \def\ZREF@xr@@input{\@input}%

```

\ZREF@xr@ignorewarning

```

908 \def\ZREF@xr@ignorewarning#1{%
909   \PackageWarningNoLine{zref/xr}{%
910     Label ‘#1’ is already in use\MessageBreak
911     in file ‘\ZREF@xr@file’%
912   }%
913   \edef\ZREF@xr@ignored{\the\numexpr\ZREF@xr@ignored+1\relax}%
914 }%

```

\ZREF@xr@checklist

```

915 \def\ZREF@xr@checklist#1#2#3\ZREF@nil{%
916   \ifx\@undefined#1\relax
917     \expandafter\ZREF@xr@checkkey\string#1\@nil
918   \fi
919   \ifx\#3\%
920   \else
921     \@ReturnAfterFi{%
922       \ZREF@xr@checklist#3\ZREF@nil
923     }%
924   \fi
925 }%
926 \long\def\@ReturnAfterFi#1\fi{\fi#1}%
927 \def\ZREF@xr@checkkey#1#2\@nil{%
928   \zref@ifpropundefined{#2}{%
929     \zref@newprop{#2}{}%
930   }{}%
931 }%

```

\ZREF@xr@scanparams

```

932 \def\ZREF@xr@scanparams#1#2#3#4#5#6#7\ZREF@nil{%
933   \global\let#1\@empty
934   \ZREF@foundfalse
935   \ZREF@xr@scantitleref#1#2\TR@TitleReference{}{}\ZREF@nil
936   \ifZREF@found
937   \else
938     \g@addto@macro#1{\default{#2}}%
939   \fi
940   % page
941   \g@addto@macro#1{\page{#3}}%
942   % nameref title
943   \ifZREF@found
944   \else
945     \ifx\#4\%
946     \else
947       \zref@ifpropundefined{title}{%
948         \zref@newprop{title}{}%
949       }{}%
950       \g@addto@macro#1{\title{#4}}%

```

```

951 \fi
952 \fi
953 % anchor
954 \ifx\#5\%
955 \else
956 \zref@ifpropundefined{anchor}{%
957 \zref@newprop{anchor}{}%
958 }{%
959 \g@addto@macro#1{\anchor{#5}}%
960 \fi
961 \ifx\#6\%
962 \else
963 \zref@ifpropundefined{url}{%
964 \zref@newprop{url}{}%
965 }{%
966 \g@addto@macro#1{\url{#6}}%
967 \fi
968 }%

```

`\ZREF@xr@scantitleref`

```

969 \def\ZREF@xr@scantitleref#1#2\TR@TitleReference#3#4#5\ZREF@nil{%
970 \ifx\#5\%
971 \else
972 \g@addto@macro#1{%
973 \default{#3}%
974 \title{#4}%
975 }%
976 \ZREF@foundtrue
977 \fi
978 }%

```

`\ZREF@xr@urlcheck`

```

979 \def\ZREF@xr@urlcheck#1{%
980 \zref@ifrefcontainsprop{#1}{anchor}{%
981 \zref@ifrefcontainsprop{#1}{url}{%
982 }{%
983 \expandafter\g@addto@macro\csname Z@R@#1\expandafter\endcsname
984 \expandafter{%
985 \expandafter\url\expandafter{\ZREF@xr@url}%
986 }%
987 }%
988 }{%
989 }%
990 }%

```

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

```

991 \define@key{ZREF@XR}{ext}{%
992 \def\zref@xr@ext{#1}%
993 }%
994 \newcommand*{\zxrsetup}{%
995 \setkeys{ZREF@XR}%
996 }%
997 \</xr>

```

6.11 Module `hyperref`

UNFINISHED :-(

```

998 \<hyperref>
999 \NeedsTeXFormat{LaTeX2e}
1000 \ProvidesPackage{zref-hyperref}%

```

```

1001 [2006/09/08 v1.3 Module hyperref for zref (H0)]
1002 \RequirePackage{zref-base}[2006/09/08]
1003 \@ifundefined{ZREF@baseok}{\endinput}{}
1004 \zref@newprop{anchor}[]{}%
1005 \@ifundefined{@currentHref}{}{\@currentHref}%
1006 }%
1007 \zref@addprop{ZREF@mainlist}{anchor}%
1008 \</hyperref>

```

6.12 Module savepos

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

6.12.1 Identification

```

1009 <*savepos>
1010 \NeedsTeXFormat{LaTeX2e}
1011 \ProvidesPackage{zref-savepos}%
1012 [2006/09/08 v1.3 Module savepos for zref (H0)]
1013 \RequirePackage{zref-base}[2006/09/08]
1014 \@ifundefined{ZREF@baseok}{\endinput}{}

```

6.12.2 Availability

First we check, whether the feature is available.

```

1015 \begingroup
1016 \@ifundefined{pdfsavepos}{%
1017 \ZREF@ErrorNoLine{%
1018 \string\pdfsavepos\space is not supported\MessageBreak
1019 in this pdfTeX version%
1020 }\ZREF@UpdatePdfTeX
1021 \endgroup
1022 \endinput
1023 }{}%
1024 \endgroup

```

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

```

1025 \RequirePackage{ifpdf}
1026 \ifpdf
1027 \else
1028 \ifnum\pdftexversion<140 %
1029 \ZREF@ErrorNoLine{%
1030 \string\pdfsavepos\space is not supported in DVI mode\MessageBreak
1031 of this pdfTeX version%
1032 }\ZREF@UpdatePdfTeX
1033 \expandafter\expandafter\expandafter\endinput
1034 \fi
1035 \fi

```

6.12.3 Setup

```

1036 \zref@newlist{savepos}
1037 \zref@newprop*{posx}[0]{\the\pdflastxpos}
1038 \zref@newprop*{posy}[0]{\the\pdflastypos}
1039 \zref@addprop{savepos}{posx}
1040 \zref@addprop{savepos}{posy}

```

6.12.4 User macros

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

```

1041 \def\zsavepos#1{%
1042   \@bsphack
1043   \if@filesw
1044     \pdfsavepos
1045     \zref@labelbylist{#1}{savepos}%
1046   \fi
1047   \@esphack
1048 }

```

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

```

1049 \newcommand*{\zposx}[1]{%
1050   \zref@extract{#1}{posx}%
1051 }%
1052 \newcommand*{\zposy}[1]{%
1053   \zref@extract{#1}{posy}%
1054 }%

```

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

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

```

1055 </savepos>

```

6.13 Module dotfill

```

1056 <*dotfill>
1057 \NeedsTeXFormat{LaTeX2e}
1058 \ProvidesPackage{zref-dotfill}%
1059   [2006/09/08 v1.3 Module dotfill for zref (H0)]
1060 \RequirePackage{zref-base}[2006/09/08]
1061 \ifundefined{ZREF@baseok}{\endinput}{\}

```

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

```

1062 \RequirePackage{zref-savepos}[2006/09/08]

```

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

```

1063 \zref@require@unique

```

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

```

1064 \RequirePackage{keyval}

```

The definitions of the keys follow.

```

1065 \define@key{ZREF@DF}{unit}{%
1066   \def\ZREF@df@unit{#1}%
1067 }
1068 \define@key{ZREF@DF}{min}{%
1069   \def\ZREF@df@min{#1}%
1070 }
1071 \define@key{ZREF@DF}{dot}{%
1072   \def\ZREF@df@dot{#1}%
1073 }

```

Defaults are set, see user interface.

```

1074 \providecommand\ZREF@df@min{2}
1075 \providecommand\ZREF@df@unit{.44em}
1076 \providecommand\ZREF@df@dot{.}

```

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

```

1077 \newcommand*{\zdotfillsetup}{\setkeys{ZREF@DF}}

```

```

\zdotfill \zdotfill sets labels at the left and the right to get the horizontal position.
\zsavepos is not used, because we do not need the vertical position.
1078 \newcommand*{\zdotfill}{%
1079   \leavevmode
1080   \global\advance\c@zref@unique\@ne
1081   \begingroup
1082     \def\ZREF@temp{zref@\number\c@zref@unique}%
1083     \pdfsavepos
1084     \zref@labelbyprops{\thezref@unique L}{posx}%
1085     \setlength{\dimen@}{\ZREF@df@unit}%
1086     \zref@ifrefundefined{\thezref@unique R}{%
1087       \ZREF@dotfill
1088     }{%
1089       \ifnum\numexpr\zposx{\thezref@unique R}-\zposx{\thezref@unique L}\relax
1090         <\dimexpr\ZREF@df@min\dimen@\relax
1091         \hfill
1092       \else
1093         \ZREF@dotfill
1094       \fi
1095     }%
1096     \pdfsavepos
1097     \zref@labelbyprops{\thezref@unique R}{posx}%
1098   \endgroup
1099   \kern\z@
1100 }

\ZREF@dotfill Help macro that actually sets the dots.
1101 \def\ZREF@dotfill{%
1102   \cleaders\hb@xt@\dimen@{\hss\ZREF@df@dot\hss}\hfill
1103 }

1104 </dotfill>

```

7 Installation

CTAN. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/zref.dtx](http://ftp.ctan.org/mirrors/macros/latex/contrib/oberdiek/zref.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/zref.pdf](http://ftp.ctan.org/mirrors/macros/latex/contrib/oberdiek/zref.pdf) Documentation.

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):

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

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

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

Refresh file databases. If your \TeX distribution (`te \TeX` , `mik \TeX` , ...) rely on file databases, you must refresh these. For example, `te \TeX` users run `texhash` or `mktexlsr`.

7.1 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 intension:

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

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

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

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

An example follows how to generate the documentation with `pdf \LaTeX` :

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

8 References

- [1] Package `everyshi`, Martin Schröder, 2001/05/15 v3.00.[CTAN:macros/latex/contrib/ms/everyshi.dtx](#)
- [2] Package `footmisc`, Robin Fairbairns, 2004/01/23 v5.3a.[CTAN:macros/latex/contrib/footmisc/footmisc.dtx](#)
- [3] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/08/16 v6.75c.[CTAN:macros/latex/contrib/hyperref/](#)
- [4] Package `lastpage`, Jeff Goldberg, 1994/06/25 v0.1b.[CTAN:macros/latex/contrib/lastpage/](#)
- [5] Package `nameref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12 v2.24.[CTAN:macros/latex/contrib/hyperref/nameref.dtx](#)
- [6] Package `perpage`, David Kastrup, 2002/12/20 v1.0.[CTAN:macros/latex/contrib/bigfoot/perpage.dtx](#)
- [7] Package `titleref`, Donald Arsenau, 2001/04/05 v3.1.[CTAN:macros/latex/contrib/misc/titleref.sty](#)
- [8] Package `totpages`, Wilhelm Müller, 1999/07/14 v1.00.[CTAN:macros/latex/contrib/totpages/](#)
- [9] Package `xr`, David Carlisle, 1994/05/28 v5.02.[CTAN:macros/latex/required/tools/xr.pdf](#)
- [10] Package `xr-hyper`, David Carlisle, 2000/03/22 v6.00beta4.[CTAN:macros/latex/contrib/hyperref/xr-hyper.sty](#)

9 History

[2006/02/20 v1.0]

- First version.

[2006/05/03 v1.1]

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

[2006/05/25 v1.2]

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

[2006/09/08 v1.3]

- Typo fixes and English cleanup by Per Starback.

10 Index

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

Symbols

`\@ReturnAfterFi` 921, 926

H		\openin 800	
\hb@xt@	1102	P	
\hfill	1091, 1102	\PackageError	153, 194, 208, 255
\hss	1102	\PackageInfo	191, 270, 807, 816
I		\PackageWarning	237, 306, 802
\if@filesw	329, 532, 1043	\PackageWarningNoLine	868, 909
\ifcase	48	\page	941
\ifcename	573, 596	\pdfastxpos	1037
\ifeof	801, 833, 856	\pdfastypos	1038
\ifnum	1028, 1089	\pdfsavepos 1018, 1030, 1044, 1083, 1096	
\ifodd	57	\pdftexversion	1028
\ifpdf	1026	\ProcessOptions	143
\ifx	160, 200,	\protect	375, 665
	222, 247, 340, 365, 404, 472,	\protected@write	334
	616, 823, 842, 846, 850, 867,	\providecommand	
	891, 916, 919, 945, 954, 961, 970		182, 771, 1074, 1075, 1076
\ifZREF@found	157, 227, 936, 943	\ProvidesPackage	
\ifZREF@immediate	319, 331, 335, 341		121, 147, 467, 494, 509, 526,
\ifzref@titleref@expand ...	640, 657		544, 559, 632, 765, 1000, 1011, 1058
\ifzref@titleref@stripperiod	653, 677	R	
\ifZREF@xr@zreflabel ..	774, 808, 841	\read	831
\immediate	324	\refstepcounter	517
\index	660	\renewcommand	443
\item	41, 44, 46, 54, 58, 60	\RequirePackage	
K			123, 128, 180, 469, 496,
\kern	1099		498, 511, 528, 546, 548, 549,
L			561, 563, 634, 636, 767, 769,
\l	302		1002, 1013, 1025, 1060, 1062, 1064
\label	472, 659, 706	\reset@font	452
\leavevmode	1079	S	
M		\section	69, 77
\m@ne	535	\setcounter	500
\mainmatter	66	\setkeys	698, 995, 1077
\makeatletter	8, 35, 778	\setlength	1085
\makeatother	33	\space	377, 818, 819, 1018, 1030
\makebox	101, 102	\stepcounter	502, 571, 572
\markboth	662	T	
\markright	664	\tableofcontents	64
\MessageBreak 803, 869, 910, 1018, 1030		\the	85,
N			87, 311, 316, 347, 361, 504, 650,
\NeedsTeXFormat			853, 863, 872, 885, 913, 1037, 1038
	120, 146, 466, 493, 508, 525,	\thetype	898
	543, 558, 631, 764, 999, 1010, 1057	\thepage	332, 336, 377, 460, 607
\newcommand		\thezpage	9, 607, 611
	12, 19, 25, 99, 471, 478, 487,	\thezref@unique 8, 443, 604, 605, 610,	
	490, 550, 578, 626, 697, 700,		611, 613, 1084, 1086, 1089, 1097
	775, 994, 1049, 1052, 1077, 1078	\title	950, 974
\newcounter	442, 499, 569	\toks@ 303, 310, 311, 316, 345,	
\newif	157, 319, 640, 653, 774		347, 360, 361, 644, 650, 840, 853
\newlabel	883, 893, 906	\TR@TitleReference	935, 969
\newpage	75	\ttl@sect@i	755
\nfss@text	452	U	
\number	28,	\unexpanded	425, 852
	43, 444, 564, 584, 608, 613, 1082	\url	966, 985
\numexpr	28,	\usepackage	4, 6
	43, 48, 584, 863, 872, 885, 913, 1089	V	
O		\verb	105
\on@line	152, 815		

W	
<code>\write</code>	323, 324
X	
<code>\x</code>	220, 221, 222, 304, 305, 307, 311, 433, 435, 583, 586, 667, 672, 839, 842, 846, 850, 864, 866, 874, 876, 886, 890
<code>\XR@ext</code>	772
Y	
<code>\y</code>	219, 222
Z	
<code>\z@</code>	579, 1099
<code>\zdotfill</code>	11, 102, 105, 1078
<code>\zdotfillsetup</code>	11, 1077
<code>\zexternaldocument</code>	11, 775
<code>\zlabel</code>	8, 17, 38, 70, 78, 471
<code>\zmakeperpage</code>	9, 578
<code>\zpageref</code>	8, 59, 487
<code>\zposx</code>	11, 85, 1049, 1089
<code>\zposy</code>	11, 87, 1049
<code>\zref</code>	8, 45, 47, 56, 61, 71, 478, 488
<code>\ZREF@@newprop</code>	277, 280
<code>\ZREF@makeperpage</code>	579, 584, 588
<code>\ZREF@newprop</code>	272, 274
<code>\ZREF@perpage@step</code>	593, 601
<code>\zref@addprop</code>	4, 10, 233, 461, 462, 505, 514, 566, 567, 568, 639, 1007, 1039, 1040
<code>\ZREF@addtoks</code>	359
<code>\ZREF@baseok</code>	463
<code>\zref@default</code>	6, 272, 449, 451
<code>\ZREF@df@dot</code>	1072, 1076, 1102
<code>\ZREF@df@min</code>	1069, 1074, 1090
<code>\ZREF@df@unit</code>	1066, 1075, 1085
<code>\ZREF@dotfill</code>	1087, 1093, 1101
<code>\ZREF@ErrorNoLine</code>	150, 170, 1017, 1029
<code>\ZREF@extract</code>	383, 388
<code>\zref@extract</code> ...	6, 29, 30, 43, 72, 382, 485, 610, 611, 707, 1050, 1053
<code>\zref@extractdefault</code>	6, 49, 50, 389, 412, 551, 613
<code>\ZREF@foundfalse</code>	218, 934
<code>\ZREF@foundtrue</code>	223, 976
<code>\zref@iflistcontainsprop</code>	5
<code>\zref@iflistundefined</code>	4, 188, 199, 207
<code>\zref@ifpropundefined</code> ...	5, 246, 254, 305, 894, 928, 947, 956, 963
<code>\ZREF@ifrefcontainsprop</code> ...	395, 403
<code>\zref@ifrefcontainsprop</code>	6, 391, 980, 981
<code>\zref@ifrefundefined</code>	6, 364, 374, 392, 413, 605, 1086
<code>\ZREF@immediatetrue</code>	322
<code>\ZREF@label</code>	292, 316, 328
<code>\zref@label</code>	5, 286, 475, 536
<code>\zref@labelbylist</code>	5, 287, 289, 604, 1045
<code>\zref@labelbyprops</code>	5, 22, 299, 1084, 1097
<code>\ZREF@listcontainsprop</code>	214, 216
<code>\zref@listcontainsprop</code>	213, 236
<code>\zref@listexists</code>	4, 206, 234, 291
<code>\ZREF@mainlist</code>	287, 455, 458, 461, 462, 505, 514, 639, 1007
<code>\ZREF@makeperpage@opt</code>	579, 581
<code>\ZREF@name</code>	149, 153, 194, 208, 237, 255, 306
<code>\zref@newlabel</code>	5, 182, 184, 354, 861, 905
<code>\zref@newlist</code> ..	4, 187, 458, 565, 1036
<code>\ZREF@newprop</code>	263, 266, 269
<code>\zref@newprop</code> ..	5, 9, 260, 459, 460, 504, 513, 564, 638, 770, 895, 929, 948, 957, 964, 1004, 1037, 1038
<code>\ZREF@nil</code>	281, 405, 420, 832, 838, 843, 847, 861, 874, 883, 890, 915, 922, 932, 935, 969
<code>\ZREF@NOVALUE</code>	411
<code>\ZREF@novalue</code>	404, 405, 411
<code>\ZREF@org@@caption</code>	714
<code>\ZREF@org@@chapter</code>	726
<code>\ZREF@org@@part</code>	720
<code>\ZREF@org@@schapter</code>	744
<code>\ZREF@org@@sect</code>	732
<code>\ZREF@org@@spart</code>	738
<code>\ZREF@org@@ssect</code>	750
<code>\ZREF@org@refstepcounter</code>	519
<code>\ZREF@org@stepcounter</code>	571, 576
<code>\ZREF@org@thepage</code>	332, 336
<code>\ZREF@org@ttl@sect@i</code>	757
<code>\ZREF@org@write</code>	323, 324
<code>\ZREF@P</code>	271, 275, 276, 277, 278, 281, 339, 341, 343, 348, 349
<code>\ZREF@patch</code>	158, 516, 711, 717, 723, 729, 735, 741, 747, 754
<code>\zref@propexists</code>	5, 235, 253, 479
<code>\zref@refused</code> ..	371, 484, 490, 554, 705
<code>\zref@require@unique</code> ..	7, 440, 570, 1063
<code>\zref@setcurrent</code> ..	5, 15, 278, 283, 518
<code>\zref@setdefault</code>	6, 448, 451
<code>\zref@setmainlist</code>	6, 454
<code>\ZREF@stripperperiod</code>	679, 684
<code>\ZREF@temp</code>	125, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 338, 345, 346, 354, 1082
<code>\ZREF@titleref</code>	701, 703
<code>\zref@titleref@cleanup</code>	642, 692
<code>\zref@titleref@current</code>	637, 638, 656, 668, 669, 674, 675, 678, 679, 695
<code>\ZREF@titleref@hook</code>	641, 645, 649, 666
<code>\zref@titleref@setcurrent</code> ..	655, 713, 719, 725, 731, 737, 743, 749
<code>\zref@titleref@stripperperiodtrue</code> ..	654
<code>\zref@titlesec@setcurrent</code>	756
<code>\ZREF@unexpanded</code>	414, 416, 425, 427, 429
<code>\ZREF@UpdatePdfTeX</code> ...	156, 1020, 1032
<code>\zref@used</code>	6
<code>\ZREF@wrapper@babel</code>	435, 437
<code>\zref@wrapper@babel</code>	7, 72, 430, 475, 480, 701
<code>\zref@wrapper@immediate</code> ..	7, 21, 320, 536
<code>\zref@wrapper@unexpanded</code>	7, 424
<code>\ZREF@X</code>	262, 265, 276

\ZREF@xr@@input	850, 907	\ZREF@xr@process@label	847, 883
\ZREF@xr@checkfile	796, 799, 827	\ZREF@xr@process@zreflabel	843, 861
\ZREF@xr@checkkey	917, 927	\ZREF@xr@processfile	799, 830, 858
\ZREF@xr@checklist	874, 915	\ZREF@xr@processline	832, 838
\zref@xr@ext	12, 771, 792, 992	\ZREF@xr@refname	862, 865, 876, 884, 887, 889, 898
\ZREF@xr@externaldocument	781, 784, 787	\ZREF@xr@scanparams	888, 932
\ZREF@xr@file	790, 800, 803, 808, 817, 825, 870, 911	\ZREF@xr@scantitleref	935, 969
\ZREF@xr@filelist	789, 823, 825, 826, 851, 852	\ZREF@xr@url	795, 985
\ZREF@xr@found	810, 818, 863, 885	\ZREF@xr@urlcheck	878, 900, 979
\ZREF@xr@graburl	792, 794	\ZREF@xr@zref@newlabel	842, 905
\ZREF@xr@ignored	811, 819, 872, 913	\ZREF@xr@zreflabelfalse	780
\ZREF@xr@ignorewarning	880, 902, 908	\ZREF@xr@zreflabeltrue	783
\ZREF@xr@line	831, 832, 843, 847	\ZREF@zref	480, 483
\ZREF@xr@list	866, 867	\zrefused	8, 26, 27, 93, 94, 95, 490
\ZREF@xr@newlabel	846, 906	\zsavepos	10, 89, 90, 1041
\ZREF@xr@prefix	788, 862, 878, 880, 884, 900, 902	\ztitleref	10, 700
\ZREF@xr@procesfile	835	\ztitlerefsetup	10, 685
		\ztotpages	9, 57, 550
		\zunmakeperpage	10, 626
		\zxrsetup	12, 991