

# The zref-vario package<sup>\*</sup>

## User manual

Gustavo Barros<sup>†</sup>

2022-12-29

### Abstract

zref-vario offers a compatibility layer for varioref to be used alongside zref-clever. It provides “\z...” counterparts to varioref’s main reference commands, each of which essentially does some (scoped) setup for varioref, then calls the original one.

### EXPERIMENTAL

## Contents

1	Loading the package	2
2	Package options	2
3	Reference commands	2
4	Reference format	3
5	Integration with zref-check	4
6	Change history	4

---

<sup>\*</sup>This file describes v0.1.6, released 2022-12-29.

<sup>†</sup><https://github.com/gusbrs/zref-vario>

## 1 Loading the package

As usual:

```
\usepackage{zref-vario}
```

zref-vario loads zref-clever, and will load varioref at begindocument/before, if it hasn't already been loaded. However, considering varioref has known load order interaction with other packages, prominently with hyperref, you may wish to load it explicitly to ensure its load order requirements are met.

## 2 Package options

Package options can be set by means of \zvsetup.

---

```
\zvsetup \zvsetup{<options>}
```

Available options are:

pageprop      varioref uses the page field of the standard label to check whether a reference and its label are on the same page, nearby, or far away. And has no real alternative data to use for the purpose. Possibly for this reason, the basic mechanism of page comparison in varioref is restricted to arabic numbered pages. Any other page numbering scheme results in varioref making no distinction between one or many pages off. But with zref it is trivial to detach the printed representation of the reference and the data used to make page comparisons. And that's what the pageprop option allows to control. It receives a zref property as value to be used by zref-vario to make page comparisons, and defaults to the page property, which corresponds to the usual varioref behavior. The property must be defined and be included in zref's main list at the time the setting is performed. The natural intended use case of this option is the abspage property provided by the zref-abspage module, which already handles these two requirements. Thus, loading this module and setting pageprop to abspage gives us page comparisons independent of the page numbering scheme, which works in the front matter with roman numbering, across front and main matter boundaries, etc. In sum, regardless of whether the page numbering is arabic or anything else.

## 3 Reference commands

---

```
\zvref \zvref{<*>[<options>]}{<label>}
```

---

```
\zvpageref \zvpageref{<*>[<options>]}{<label>}
```

---

```
\zvrefrange \zvrefrange{<*>[<options>]}{<label>}{<label>}
```

---

<code>\zvpagerefrange</code>	<code>\zvpagerefrange&lt;*&gt;[&lt;options&gt;]{&lt;label&gt;}{&lt;label&gt;}</code>
------------------------------	--

---

<code>\zfullref</code>	<code>\zfullref&lt;*&gt;[&lt;options&gt;]{&lt;label&gt;}</code>
------------------------	---

---

These user commands work pretty much as their `varioref` counterparts.<sup>1</sup> Indeed, they are just wrappers around them. As such, differently from `zref-clever`'s commands, these can receive single labels as arguments, not lists of them. In all of them the starred version inhibits hyperlinking, and they (locally) set `varioref`'s `nospace` option, so that the this syntax can be used unambiguously.

All of them have a single optional argument, which can receive any of `\zceref`'s options, and those are set for calls to `\zceref` or `\zpageref` which are part of the building blocks of the reference formats and strings set for `varioref`. However, there is indeed some potential friction and caveats stemming from the use of these options, which were designed to work for single calls of `\zceref`, to apply to *pairs* of them. Hence, whether each and every of these options is meaningful, useful, or potentially harmful in this context depends on the case, and it is up to the user to make proper use of them. Of course, one can always split the pair using `\zceref` and then `\zvpageref`, `\zvpagerefrange`, or `\zcpageref`, and have more control over each part. Anyway, the package does what it says: passes the options along to the underlying call(s) to `\zceref`.<sup>2</sup>

<code>vcurrent</code>	Besides <code>\zceref</code> 's options, mentioned above, two other ones are provided for <code>zref-</code>
<code>vother</code>	<code>vario</code> 's reference commands, corresponding to <code>varioref</code> 's commands optional arguments: <code>vcurrent</code> (the first optional argument) and <code>vother</code> (the second optional argument). Of course, these are only available when the underlying <code>varioref</code> command supports them.

## 4 Reference format

---

<code>\zvLanguageSetup</code>	<code>\zvLanguageSetup{&lt;language&gt;}{&lt;options&gt;}</code>
-------------------------------	--

---

User interface for customization of “strings” and “formats” for `<language>`. `<language>` must be known to `zref-clever`. The `<options>` are familiar to users of `varioref`: `retextfaceafter`, `retextfacebefore`, `retextafter`, `retextbefore`, `retextcurrent`, `retextfaraway`, `retextpagerange`, `retextlabelrange`, `vrefformat`, `vrefrangeformat`, and `fullrefformat`.<sup>3</sup> Their meaning is the same as the ones they have in `varioref` and, indeed, they work by setting those `varioref` macros to the values given to the corresponding options in `\zvLanguageSetup`. If you are setting up a language which has no built-in support, you should set at least the whole set of “`retext...`” options. Language independent default values are provided for the “`...format`” options (equivalent to the ones from `varioref`, which are adequate for most use cases), so you may omit them.

<sup>1</sup>If you are not acquainted with them, see `varioref`'s documentation.

<sup>2</sup>Note that the options given to each call of these user macros are set *before* the call to the original `varioref` counterpart. This means that options given to `\zceref` or `\zpageref` in the strings and formats in `\zvLanguageSetup` will have precedence over the former. Which is useful, so that we can distinguish to some degree the first from the second call of `\zceref`/`\zpageref` done in the formats, but does not eliminate the limitations which arise from the underlying problem.

<sup>3</sup>For details, see `varioref`'s documentation.

But, if you need to adjust them, the default values can be overridden by setting the corresponding options in `\zvLanguageSetup`.

---

<code>\zvhyperlink</code>	<code>\zvhyperlink{&lt;text&gt;}</code>
---------------------------	---

---

Inside the options of `\zvLanguageSetup`, and also inside the `vcurrent` and `vother` options of `zref-vario`'s reference commands, `\zvhyperlink` can be used to produce a hyperlink to the label of the reference. For example, one could set `reftextafter={on the \zvhyperlink{next page}}` to get "next page" hyperlinked. `\zvhyperlink` can be used regardless of `hyperref` being enabled for `zref-clever`, it will just pass on the `<text>` argument if it is not. On range reference commands, `\zvhyperlink` only produces a link when both labels fall on the same page, in which case it hyperlinks to the first label.

As far as `zref-vario` is concerned, `varioref`'s options are (mostly) not taken into account: the language options are disregarded (settings made with `\zvLanguageSetup` are used instead), the `nospace` option is hard-coded (locally) for the "`\z...`" commands, and `draft` and `final` are typically given to `\documentclass`, though they do affect `zref-vario`'s commands, just as they do with `varioref`'s.

## 5 Integration with zref-check

When package `zref-check` is loaded, `zref-vario` provides one further option to its user commands: `vcheck`. The purpose of this option is to check the relative position of label and reference within the same page. It can receive two values: `above` and `below`, being those the names of the `zref-check`'s checks which are meaningful for the use case at hand. Of course, these could also be performed with `\zceref`'s `check` option, which is available for `zref-vario`'s commands as well. The difference here is that the check specified in `vcheck` is only performed when the referenced label – or labels, in the case of a range – fall on the same page as the reference itself.<sup>4</sup> In other words, when `vcurrent` would be used (if provided). For this reason, only the commands which support `vcurrent` offer `vcheck`. Also, `vcheck` cannot receive `zref-check`'s options, as `check` is able to, but the latter can be used for the purpose of locally controlling the behavior of the checks performed by `vcheck`. Consult `zref-check`'s documentation for details and limitations of these checks and envisaged workflows for their reliable use.

## 6 Change history

A change log with relevant changes for each version, eventual upgrade instructions, and upcoming changes, is maintained in the package's repository, at <https://github.com/gusbrs/zref-vario/blob/main/CHANGELOG.md>. The change log is also distributed with the package's documentation through CTAN upon release so, most likely, `texdoc zref-vario/changelog` should provide easy local access to it. An

---

<sup>4</sup>There's another technical difference between them. `zref-check`'s checks, and hence those of the `check` option, make sure the whole reference passes each check by setting labels both at the start and the end of the reference, and verifying if each one of them passes the checks. But, since `varioref` already has its own mechanism to handle references which cross page boundaries, `vcheck`'s checks set only one label, at the end of the reference, the same position `varioref` uses to check whether label and reference are on the same page.

archive of historical versions of the package is also kept at <https://github.com/gusbrs/zref-vario/releases>.