zebra — Writing Revision Toolkit*

Ruini Xue†

v1.8.0 (2026/04/26)

Abstract

The zebra package is a writing revision toolkit. The current release focuses on inline note-taking, with a lightweight set of macros designed to be simple and practical for both solo and collaborative workflows. Five built-in commands—`\todo`, `\note`, `\comment`, `\fixed`, and `\placeholder`—cover common use cases out of the box, and `\zebranewnote` lets you define additional note types as needed. Notes are automatically numbered per type, marked with a customisable symbol (default: `\textdbend`) in the nearest margin, and summarised with a summary table plus a detailed note list at the end of the document. Passing the `final` option suppresses all notes for production output.

Contents

*This package was previously distributed as zebra-goodies. The old name still works but will print a deprecation warning. Please update to zebra.

†Email: xueruini@gmail.com

1 Introduction

zebra is a writing revision toolkit. The current release focuses on inline note-taking. Many note-taking and to-do packages exist for LaTeX, but most fall into one of two traps: they either offer an overwhelming feature set that tries to cover every conceivable use case, or they clutter the margins with oversized colourful boxes and arrows that make the document hard to read.

zebra takes a different approach. It aims to be *simple* —intuitive commands with only the arguments you actually need—and *good enough*—notes appear inline with a small visual cue in the margin, keeping the document readable while still making annotations easy to spot. Each note type is automatically numbered, and a summary table plus a detailed note list at the end of the document serve as a gentle reminder to address them before the final version.

2 Installation

zebra is available on CTAN; install it through your LaTeX distribution's package manager. To build from source, run `latexmk zebra.dtx` to extract the package and typeset the documentation in one step.

3 Using the package

Load the package in the preamble with any desired options.

```
\usepackage[<options>]{zebra}  % was zebra-goodies
```

3.1 Package Options

`draft` `final` These two options are complementary. Default: `true` (draft mode). All notes are typeset inline and a summary table plus a detailed note list are appended at the end of the document. Setting `final` (or `draft=false`) suppresses all notes and the generated lists, producing clean output ready for distribution.

`sort` Controls the order of the detailed note list printed at the end of the document. Default: `none` (document order). `sort=type` groups them by note type.

`pagelinks` Controls whether page numbers in the detailed note list are clickable. Default: `true`. Set `pagelinks=false` to disable these links. The complementary option `nopagelinks` is also accepted.

`font-expansion` Controls microtype font expansion. This usually improves the appearance of the document. Disable it if it conflicts with your engine or another package by setting `font-expansion=false`. The complementary option `nofont-expansion` is also accepted. Default: `true`. microtype remains loaded when expansion is disabled.

`unnumbered` Turns off note numbering. Inline markers become `[todo:  ...]` (no number), the margin symbol carries no number, the end-of-document detailed note list is omitted (the per-type summary table remains), and `\zebraref` falls back to `\ref`. Useful as a workaround if numbered notes interact badly with a particular class; the summary table count is then an upper-bound estimate. Default: `false`.

3.2 Notes Macros

All note commands share the syntax `\cmd[⟨name⟩]{⟨text⟩}`. Each also has a prefixed alias (e.g. `\zebratodo`) that is always available, regardless of name conflicts. If a short name clashes with another loaded package, zebra will *not* overwrite the existing definition; use the prefixed form instead.

`\todo`	`\todo[⟨name⟩]{⟨text⟩}`
`\zebratodo`	`\zebratodo[⟨name⟩]{⟨text⟩}`

The primary command provided by zebra is `\todo`. It inserts an inline note in the current paragraph, typeset in a predefined colour and marked with a symbol in the nearest margin. The mandatory ⟨*text*⟩ describes the task; the optional ⟨*name*⟩ specifies who is responsible for addressing it, which is particularly useful during collaborative writing.

```
The motivation section still feels too vague \todo{revise the introduction
before submission} and could benefit from a concrete running example to
guide the reader through the key ideas step by step.
```

The motivation section still feels too vague and could benefit from a concrete running example to guide the reader through the key ideas step by step.

The optional argument assigns one or more people to the note. Assignees appear prefixed with `@`, and notes of the same type are numbered sequentially.

```
The related work section needs more references \todo[alice]{add two or
three citations from the latest survey} to recent advances in the field. We
should also double-check the experimental setup before the camera-ready
deadline \todo[bob, carol]{verify the hyperparameter table against the
source code and update any outdated entries and let's check afterwards}.
```

The related work section needs more references [TODO 1@alice: add two or three citations from the latest survey] to recent advances in the field. We should also double-check the experimental setup before the camera-ready deadline [TODO 2@bob, carol: verify the hyperparameter table against the source code and update any outdated entries and let's check afterwards].

Notes can appear inside moving arguments such as `\section` and `\caption`. To ensure stable numbering and cross-references, add a `\label` inside the note:

```
\section{Introduction\todo[jerry]{\label{zebra:heading}fix the name}}
\begin{figure}
  \caption{Speed vs distance. \todo{need to insert the figure}}
\end{figure}
```

With a `\label`, the note is counted once regardless of how many times the heading appears (table of contents, running headers, etc.). Notes without a `\label` in moving arguments are still safe but may receive a separate number in each context.

`\note`	`\note[⟨name⟩]{⟨text⟩}`
`\zebranote`	`\zebranote[⟨name⟩]{⟨text⟩}`

`\comment`	`\comment[⟨name⟩]{⟨text⟩}`
`\zebracomment`	`\zebracomment[⟨name⟩]{⟨text⟩}`

\fixed	\fixed[⟨name⟩]{⟨text⟩}
\zebrafixed	\zebrafixed[⟨name⟩]{⟨text⟩}

\placeholder	\placeholder[⟨name⟩]{⟨text⟩}
\zebraplaceholder	\zebraplaceholder[⟨name⟩]{⟨text⟩}

These commands share the same syntax and behaviour as \todo; they differ only in name and colour, providing semantic distinction for different annotation purposes. Note that \zebracomment is used in the example below because \comment is already defined by l3doc.

```
We may want to reorganise \note{how should we structure the intro?} this
part before the final submission. The experimental setup in Section~2 has
already been reviewed by a collaborator \zebracomment[tom]{the setup
description looks clear now}. Results are presented in the following tables
and figures, but some of them are still missing.

The discussion has been revised \placeholder[lucy, tom]{good job!}
and the related work comparison strengthened with two additional references.
The list of references still needs a second pass \todo{check bibliography
entries for formatting} before we can finalize the submission.

With those items addressed, the conclusion has been rewritten so the
argument flows more naturally from the results. \fixed[John]{updated the
conclusion} The overall structure now matches the revised outline we agreed
on last week. \note[who]{anything else?} If not, the draft should be fine.
```

We may want to reorganise [Note 1: how should we structure the intro?] this part before the final submission. The experimental setup in Section 2 has already been reviewed by a collaborator [Comment 1@tom: the setup description looks clear now]. Results are presented in the following tables and figures, but some of them are still missing.
The discussion has been revised [Placeholder 1@lucy, tom: good job!] and the related work comparison strengthened with two additional references. The list of references still needs a second pass [Todo 3: check bibliography entries for formatting] before we can finalize the submission. With those items addressed, the conclusion has been rewritten so the argument flows more naturally from the results. [Fixed 1@John: updated the conclusion] The overall structure now matches the revised outline we agreed on last week. [Note 2@who: anything else?] If not, the draft should be fine.

\zebranewnote	\zebranewnote{⟨note name⟩}{⟨xcolor name⟩}[⟨symbol⟩]

Creates a new note type. The ⟨*note name*⟩ becomes the command name (e.g. passing question creates \question and \zebraquestion), and ⟨*xcolor name*⟩ sets its colour. The colour must be a named colour already known to xcolor; define it with \definecolor or \colorlet beforehand if needed. The optional ⟨*symbol*⟩ overrides the default margin symbol (\textdbend) for this note type only. Per-type symbols can also be changed after loading via \zebrasetup{symbol/⟨*type*⟩=⟨*symbol*⟩}.

```
\colorlet{mycyan}{cyan!80!black}
\zebranewnote{question}{mycyan}[\faQuestionCircle] % \usepackage{fontawesome}

When it moves to the next step, we should be fine.\question[who]{what's this?}
```

When it moves to the next step, we should be fine.[Question 1@who: what's this?]

`\zebraref` `\zebraref{⟨label⟩}`

Labels may be placed inside note bodies with the usual `\label` command. Standard `\ref` returns the note number, while `\zebraref` prints the note type together with the number.

```
The motivation section still feels too vague \todo{\label{zebra:intro}revise
the introduction before submission}.  The same issue appears again later
\note{see Todo~\ref{zebra:intro} (that is, \zebraref{zebra:intro}) on
p.~\pageref{zebra:intro}}.
```

The motivation section still feels too vague [Todo 4: revise the introduction before submission]. The same issue appears again later [Note 3: see Todo 4 (that is, Todo 4) on p. 5].

As in standard LaTeX, labels inside notes are unavailable in `final` mode because the notes themselves are suppressed.

`\zebrasetup` `\zebrasetup{⟨key = value list⟩}`

Configures note appearance after loading. Accepted keys:

- `color/`⟨*type*⟩`=`⟨*colour*⟩ — override the colour of a note type.
- `symbol/`⟨*type*⟩`=`⟨*symbol*⟩ — override the margin symbol of a note type.

For example:

```
\zebrasetup{symbol/fixed=\manerrarrow} % like this doc
\zebrasetup{color/todo=red}
```

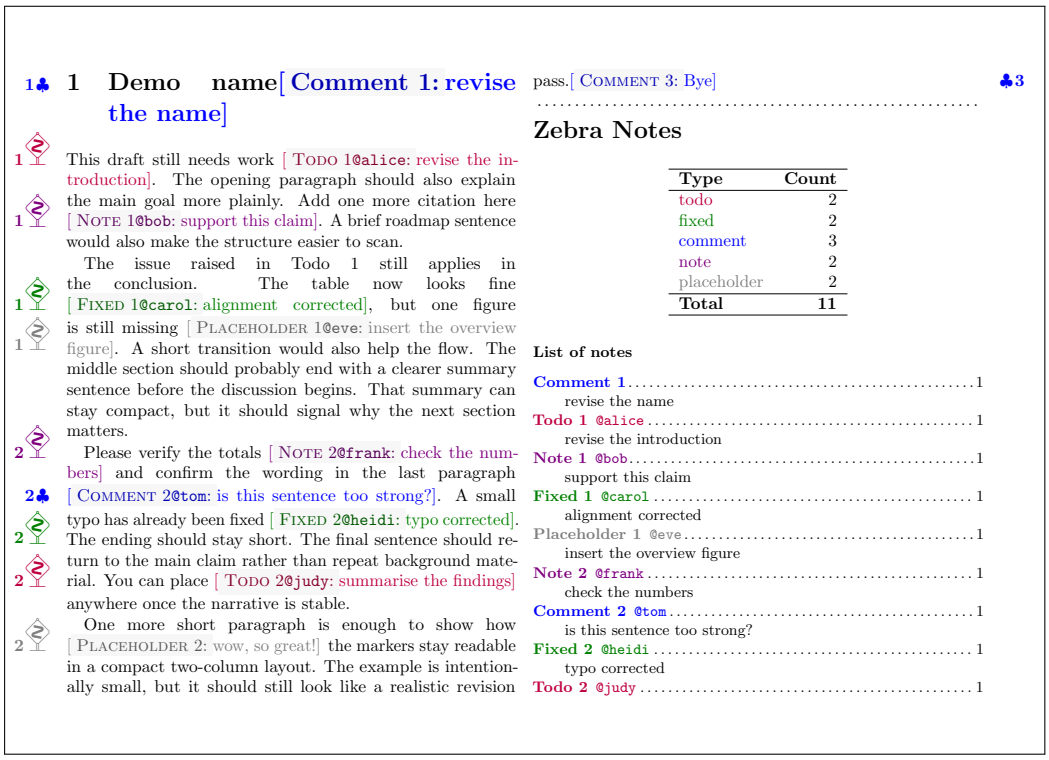
3.3 Two-column Support

In `twocolumn` documents, the margin symbol is automatically placed on the nearest margin: left margin for the left column, right margin for the right column. No special configuration is needed. This also works correctly in combination with the `twoside` option.

```
\usepackage[paperwidth=21cm,paperheight=15cm,margin=1.1cm]{geometry}
\usepackage{zebra}
\zebrasetup{symbol/comment=$\clubsuit$}
\pagestyle{empty}
\begin{document}
\section{Demo name\comment{revise the name}}
This draft still needs work
\todo[alice]{\label{zebra:intro}revise the introduction}. The
opening paragraph should also explain the main goal more plainly.
Add one more citation here \note[bob]{support this claim}. A
brief roadmap sentence would also make the structure easier to
scan.

The issue raised in Todo~\ref{zebra:intro} still applies in
the conclusion. The table now looks fine
\fixed[carol]{alignment corrected}, but one figure is still
missing \placeholder[eve]{insert the overview figure}. A short
transition would also help the flow. The middle section should
```

```
probably end with a clearer summary sentence before the
discussion begins. That summary can stay compact, but it should
signal why the next section matters.

Please verify the totals \note[frank]{check the numbers} and
confirm the wording in the last paragraph
\comment[tom]{is this sentence too strong?}. A small typo has
already been fixed \fixed[heidi]{typo corrected}. The ending
should stay short. The final sentence should return to the main
claim rather than repeat background material. You can place
\todo[judy]{summarise the findings} anywhere once the narrative
is stable.

One more short paragraph is enough to show how \placeholder{wow,
so great!} the markers stay readable in a compact two-column layout.
The example is intentionally small, but it should still look like
a realistic revision pass.\comment{Bye}
\end{document}
```

The code above produces the following output:



1♣ **1 Demo name[Comment 1: revise the name]**

1 This draft still needs work [Todo 1@alice: revise the introduction]. The opening paragraph should also explain the main goal more plainly. Add one more citation here 1 [Note 1@bob: support this claim]. A brief roadmap sentence would also make the structure easier to scan.

The issue raised in Todo 1 still applies in the conclusion. The table now looks fine 1 [Fixed 1@carol: alignment corrected], but one figure 1 is still missing [Placeholder 1@eve: insert the overview figure]. A short transition would also help the flow. The middle section should probably end with a clearer summary sentence before the discussion begins. That summary can stay compact, but it should signal why the next section matters.

2 Please verify the totals [Note 2@frank: check the numbers] and confirm the wording in the last paragraph 2♣ [Comment 2@tom: is this sentence too strong?]. A small typo has already been fixed 2 [Fixed 2@heidi: typo corrected]. The ending should stay short. The final sentence should return to the main claim rather than repeat background material. 2 You can place [Todo 2@judy: summarise the findings] anywhere once the narrative is stable.

One more short paragraph is enough to show how 2 [Placeholder 2: wow, so great!] the markers stay readable in a compact two-column layout. The example is intentionally small, but it should still look like a realistic revision pass.[Comment 3: Bye] ♣3

Zebra Notes

Type	Count
todo	2
fixed	2
comment	3
note	2
placeholder	2
Total	**11**

List of notes

4 Implementation

1 ⟨*package⟩

2 ⟨@@=zebra⟩

Version data to start with.

```
\ProvidesExplPackage{zebra}
  {2026/04/26}
  {1.8.0}
  {Writing Revision Toolkit}
```

4.1 Package options

Package options `draft`, `pagelinks`, `font-expansion`, `sort`, and `unnumbered` are created using the kernel key–value interface available since LaTeX 2022-06-01. Post-load configuration (`\zebrasetup`) uses a separate `zebra-setup` key family with `color/`⟨*type*⟩ and `symbol/`⟨*type*⟩ sub-families.

```
\bool_new:N \l__zebra_draft_bool
\bool_new:N \l__zebra_microtype_expansion_bool
\bool_new:N \l__zebra_pagelinks_bool
\bool_new:N \l__zebra_sort_none_bool
\bool_new:N \l__zebra_unnumbered_bool
\seq_new:N \g__zebra_note_types_seq
\prop_new:N \g__zebra_note_colors_prop
\prop_new:N \g__zebra_note_public_alias_prop
\int_new:N \g__zebra_note_id_int
\tl_new:N \l__zebra_note_target_tl
\tl_new:N \l__zebra_note_color_tl
\tl_new:N \l__zebra_note_ref_type_tl
\tl_new:N \l__zebra_summary_rows_tl
\int_new:N \l__zebra_total_notes_int
\prop_new:N \g__zebra_note_symbols_prop
\tl_new:N \l__zebra_symbol_tl
\tl_set:Nn \l__zebra_symbol_tl { \textdbend }

\msg_new:nnn { zebra } { command-taken }
  {
    The~command~'\iow_char:N\\#1'~is~already~defined.~
    Use~'\iow_char:N\\zebra#1'~instead.
  }
\msg_new:nnn { zebra } { invalid-note-label }
  { Label~'#1'~is~not~a~zebra~note~label. }
\msg_new:nnn { zebra } { duplicate-note-label }
  { Note~label~'#1'~used~by~a~different~note;~second~note~gets~
    its~own~identity. }

\prg_new_conditional:Npnn \__zebra_if_package_loaded:n #1 { T , F , TF }
  {
    \cs_if_exist:cTF { ver@#1.sty }
      { \prg_return_true: }
      { \prg_return_false: }
  }

\keys_define:nn { zebra }
  {
    draft .bool_set:N = \l__zebra_draft_bool,
    draft .initial:n = true,
    final .meta:n = { draft = false },
    font-expansion .bool_set:N = \l__zebra_microtype_expansion_bool,
    font-expansion .initial:n = true,
```

```
    nofont-expansion .meta:n = { font-expansion = false },
    pagelinks .bool_set:N = \l__zebra_pagelinks_bool,
    pagelinks .initial:n = true,
    nopagelinks .meta:n = { pagelinks = false },
    sort .choice:,
    sort / type .code:n = { \bool_set_false:N \l__zebra_sort_none_bool },
    sort / none .code:n = { \bool_set_true:N \l__zebra_sort_none_bool },
    sort .initial:n = none,
    unnumbered .bool_set:N = \l__zebra_unnumbered_bool,
    unnumbered .initial:n = false,
  }
\ProcessKeyOptions [ zebra ]
\keys_define:nn { zebra-setup / color }
  {
    unknown .code:n =
      { \prop_gput:NVn \g__zebra_note_colors_prop \l_keys_key_str {#1} }
  }
\keys_define:nn { zebra-setup / symbol }
  {
    unknown .code:n =
      { \prop_gput:NVn \g__zebra_note_symbols_prop \l_keys_key_str {#1} }
  }
\sys_if_engine_xetex:T
  { \bool_set_false:N \l__zebra_microtype_expansion_bool }

\cs_new_protected:Npn \__zebra_setup_microtype:
  {
    \__zebra_if_package_loaded:nTF { microtype }
      {
        \bool_if:NF \l__zebra_microtype_expansion_bool
          { \microtypesetup { expansion = false } }
      }
      {
        \bool_if:NTF \l__zebra_microtype_expansion_bool
          { \RequirePackage{microtype} }
          { \RequirePackage[expansion=false]{microtype} }
      }
  }
\__zebra_setup_microtype:
\__zebra_if_package_loaded:nF { hyperref }
  {
    \bool_if:NT \l__zebra_pagelinks_bool
      {
        \RequirePackage{hyperref}
        \hypersetup { pdfborder = { 0~0~0 } }
      }
  }
```

4.2 Moving-argument deduplication

Notes inside moving arguments (`\section`, `\caption`, etc.) may be processed more than once per compilation pass. Two separate problems are handled independently:

Problem A — `\sbox` re-measurement. `\@makecaption` typesets the caption in an `\sbox` for width measurement, then typesets it again if it is long. Both executions share the same `\inputlineno`, so the *instance key* (⟨*type*⟩|⟨*author*⟩|⟨*body*⟩|`\inputlineno`) catches the replay. The second execution reuses the first's allocation and re-renders, so writes that were lost inside the discarded `\sbox` are re-emitted by the actual typesetting pass.

Problem B — TOC/LOF/header replay. The note token is written verbatim to `.toc`/`.lof`/marks and re-executed in a secondary context with a different `\inputlineno`. For *labeled* notes the replay is caught by two mechanisms:

1. A *stable key* (⟨*type*⟩|⟨*label name*⟩) stored alongside the allocation; a later encounter from marks/headers that still carries the `\label` finds this key and suppresses.

2. A *content signature* (⟨*type*⟩|⟨*author*⟩|⟨*sanitised body*⟩) written to the `.aux` file; on the next pass, TOC/LOF encounters whose `\label` was consumed by `\protected@write`'s `\edef` match the signature and suppress.

Unlabeled notes in moving arguments receive independent allocations (cosmetic duplicate); adding `\label` is the recommended fix.

```
\RequirePackage{xcolor}
\RequirePackage{marginnote}
\cs_new_eq:NN \__zebra_kernel_label:n \label
%% -- dedup data structures --
%% Maps any key (instance, stable, or content-sig) to the allocation.
\prop_new:N \g__zebra_note_target_prop
\prop_new:N \g__zebra_note_display_prop
%% Content signatures of labeled notes from previous pass (.aux).
\prop_new:N \g__zebra_note_sig_known_prop
%% Content signatures written this pass (dedup aux writes).
\prop_new:N \g__zebra_note_sig_written_prop
%% Content signature stored per stable key (for label-conflict detection).
\prop_new:N \g__zebra_note_stable_sig_prop
\tl_new:N \l__zebra_note_display_tl
\tl_new:N \l__zebra_note_key_tl
%% Instance key: unique per source location.
\cs_new:Npn \__zebra_instance_key:nnn #1#2#3
  {
    \tl_to_str:n {#1}
    | \tl_to_str:n {#2}
    | \tl_to_str:n {#3}
    | \int_eval:n { \tex_inputlineno:D }
  }
%% Content signature: body stringified with ALL \label{...} stripped.
%% Matches across body (has labels), TOC (labels consumed by \edef),
%% and running heads (which may uppercase the author/body text).
%% The optional \protect prefix covers marks and \protected@write paths.
%% Uses replace_all so that multiple labels are all stripped.
\cs_new_protected:Npn \__zebra_content_sig:nnnN #1#2#3#4
  {
    \tl_set:Nx \l_tmpa_tl { \tl_to_str:n {#3} }
    \regex_replace_all:nnN
```

```
      { (?: \\protect \s* )? \\label \s* \{ [^{}]* \} } { } \l_tmpa_tl
    \tl_set:Nx #4
      {
        \tl_to_str:n {#1}
        | \str_lowercase:f { \tl_to_str:n {#2} }
        | \str_lowercase:f { \l_tmpa_tl }
      }
  }
%% Extract the first \label name from the stringified body.
%% Sets #2 to the label name, or clears it if none found.
%% The optional \protect prefix covers marks and \protected@write paths.
\cs_new_protected:Npn \__zebra_extract_label:nN #1#2
  {
    \tl_set:Nx \l_tmpb_tl { \tl_to_str:n {#1} }
    \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl
    \regex_replace_once:nnN
      { \A .* (?: \\protect \s* )? \\label \s* \{ ([^{}]*) \} .* \Z }
      { \1 } \l_tmpb_tl
    \tl_if_eq:NNTF \l_tmpb_tl \l_tmpc_tl
      { \tl_clear:N #2 }
      { \tl_set_eq:NN #2 \l_tmpb_tl }
  }
%% Stable key for labeled notes.
%% #2 is expected to be already stringified (from regex extraction),
%% so no \tl_to_str is applied - otherwise an unexpanded variable
%% token would be stringified instead of its value.
\cs_new:Npn \__zebra_stable_key:nn #1#2
  { \tl_to_str:n {#1} | label | #2 }
%% Allocate a fresh note: increment the type counter, generate
%% a unique hypertarget name, and record the note in the list body.
\cs_new_protected:Npn \__zebra_allocate_note:nnn #1#2#3
  {
    \int_gincr:c { g__zebra_note_count_#1_int }
    \tl_set:Nx \l__zebra_note_display_tl { \__zebra_note_count:n {#1} }
    \int_gincr:N \g__zebra_note_id_int
    \tl_set:Nx \l__zebra_note_target_tl
      { zebranote.\int_use:N \g__zebra_note_id_int }
    \__zebra_record_note:nnnnn
      {#1}
      { \l__zebra_note_display_tl }
      {#2}
      {#3}
      { \l__zebra_note_target_tl }
  }
%% Aux-file interface: record a content signature together with the
%% originating instance key.  A later encounter whose instance key
%% differs from the stored one is a replay and is suppressed.
%% Re-stringify for catcode normalisation.
\cs_new_protected:Npn \zebra@sig #1#2
  {
    \tl_set:Nx \l_tmpa_tl { \tl_to_str:n {#1} }
    \tl_set:Nx \l_tmpb_tl { \tl_to_str:n {#2} }
    \prop_gput:NVV \g__zebra_note_sig_known_prop \l_tmpa_tl \l_tmpb_tl
  }
```

```
\cs_new_protected:Npn \__zebra_write_sig:NN #1#2
  {
    \prop_if_in:NVF \g__zebra_note_sig_written_prop #1
      {
        \immediate\write \@auxout
          { \string\zebra@sig { \tl_use:N #1 } { \tl_use:N #2 } }
        \prop_gput:NVn \g__zebra_note_sig_written_prop #1 { 1 }
      }
  }
\cs_if_exist:NTF \dbend
  {
    \cs_set_eq:NN \__zebra_saved_dbend: \dbend
    \cs_undefine:N \dbend
    \RequirePackage{manfnt}
    \cs_set_eq:NN \dbend \__zebra_saved_dbend:
  }
  { \RequirePackage{manfnt} }
\cs_new_protected:Npn \__zebra_pdfstring_note:
  { \@ifnextchar [ { \__zebra_pdfstring_note_opt:w } { \use_none:n } }
\cs_new_protected:Npn \__zebra_pdfstring_note_opt:w [#1] #2 { }
\cs_new:Npn \__zebra_target:nn #1#2 {#2}
\cs_new:Npn \__zebra_link:nn #1#2 {#2}
\cs_new:Npn \__zebra_pageref:n #1 { \pageref {#1} }
\cs_new:Npn \__zebra_zebra_label_name:n #1 { #1@zebra }
\cs_new:Npn \__zebra_zebra_label_type:n #1
  {
    \exp_after:wN \use_i:nn
      \cs:w r@\__zebra_zebra_label_name:n {#1}\cs_end:
      { }
  }
\cs_new_protected:Npn \__zebra_write_zebra_label:n #1
  {
    \protected@write \@auxout { }
      {
        \string\newlabel{\__zebra_zebra_label_name:n {#1}}
          {{\exp_not:V \l__zebra_note_ref_type_tl}{}}
      }
  }
\cs_new_protected:Npn \__zebra_note_label:n #1
  {
    \__zebra_kernel_label:n {#1}
    \__zebra_write_zebra_label:n {#1}
  }
\cs_new_protected:Npn \__zebra_zebra_ref:n #1
  {
    \cs_if_exist:cTF { r@\__zebra_zebra_label_name:n {#1} }
      { \__zebra_zebra_label_type:n {#1}~\ref{#1} }
      {
        \msg_warning:nnn { zebra } { invalid-note-label } {#1}
        ??
      }
  }
\NewDocumentCommand \zebraref { m }
  {
```

```
    \bool_if:NTF \l__zebra_unnumbered_bool
      { \ref{#1} }
      { \__zebra_zebra_ref:n {#1} }
  }
\cs_new_protected:Npn \__zebra_apply_pdfstring_defs:
  {
    \pdfstringdefDisableCommands
      {
        \cs_set:Npn \zebraref ##1 { \ref{##1} }
        \seq_map_inline:Nn \g__zebra_note_types_seq
          {
            \cs_set_eq:cN { zebra##1 } \__zebra_pdfstring_note:
            \prop_if_in:NnT \g__zebra_note_public_alias_prop { ##1 }
              { \cs_set_eq:cN { ##1 } \__zebra_pdfstring_note: }
          }
      }
  }
\cs_new_protected:Npn \__zebra_setup_pagelinks:
  {
    \cs_set:Npn \__zebra_target:nn ##1##2 {##2}
    \cs_set:Npn \__zebra_link:nn ##1##2 {##2}
    \cs_set:Npn \__zebra_pageref:n ##1 { \pageref {##1} }
    \__zebra_if_package_loaded:nT { hyperref }
      {
        \cs_set:Npn \__zebra_pageref:n ##1 { \pageref* {##1} }
        \bool_if:NT \l__zebra_pagelinks_bool
          {
            \cs_set:Npn \__zebra_target:nn ##1##2 { \hypertarget{##1}{##2} }
            \cs_set:Npn \__zebra_link:nn ##1##2 { \hyperlink{##1}{##2} }
          }
        \__zebra_apply_pdfstring_defs:
      }
  }
\hook_gput_code:nnn { begindocument } { zebra }
  { \__zebra_setup_pagelinks: }
```

4.3 Main notes macros

Various helper macros are defined before reaching out to the `\todo` commands.

Place the margin note on the nearest margin. Takes two arguments: `#1` for the left margin (number then symbol) and `#2` for the right margin (symbol then number), so the symbol always sits closest to the text column. In single-column mode `\marginnote` is used with the right-margin variant as default. In twocolumn mode `\marginpar` positions the symbol close to the text column reliably (it picks side from `\if@firstcolumn`, not from a `.aux` round trip), so we keep that path for the body. We must, however, route around the contexts where `\marginpar` would crash with "Float(s) lost": caption sboxes (inner mode), caption parboxes (signalled by `\@captype`), and the wide `\vbox` that `\@topnewpage` builds for `\title` (signalled by `\hsize` > `\columnwidth`). Those contexts fall back to `\marginnote`, where we additionally replicate `\@marginparreset` (`\@parboxrestore` + `\normalfont\normalsize`) so the marker does not inherit a `\Huge` font from `\title`.

```
\cs_new_protected:Npn \__zebra_margin_note:nn #1#2
  {
    \legacy_if:nTF { @twocolumn }
```

```
      {
        \bool_lazy_or:nnTF
          { \mode_if_inner_p: }
          {
            \bool_lazy_or_p:nn
              { \cs_if_exist_p:N \@captype }
              { \dim_compare_p:nNn { \hsize } > { \columnwidth } }
          }
          {
            \marginnote
              [ { \@parboxrestore \normalfont \normalsize #1 } ]
              { \@parboxrestore \normalfont \normalsize #2 }
          }
          {
            \marginpar
              [ { \makebox[\marginparwidth][r]{#1} } ]
              { \makebox[\marginparwidth][l]{#2} }
          }
      }
      {
        \marginnote
          [ { \@parboxrestore \normalfont \normalsize #1 } ]
          { \@parboxrestore \normalfont \normalsize #2 }
      }
  }
\cs_new:Npn \__zebra_prepend:nn #1#2
  { \tl_if_blank:nTF {#2} {} {#1#2} }
\cs_new:Npn \__zebra_capitalize_type:n #1
  { \text_uppercase:n { \tl_head:n {#1} } \tl_tail:n {#1} }
\cs_new:Npn \__zebra_note_count:n #1
  { \int_use:c { g__zebra_note_count_#1_int } }
\cs_new:Npn \__zebra_note_color:n #1
  { \prop_item:Nn \g__zebra_note_colors_prop {#1} }
\cs_new:Npn \__zebra_note_symbol:n #1
  {
    \prop_if_in:NnTF \g__zebra_note_symbols_prop {#1}
      { \prop_item:Nn \g__zebra_note_symbols_prop {#1} }
      { \l__zebra_symbol_tl }
  }
\cs_new_protected:Npn \__zebra_new_listbody:n #1
  { \tl_new:c { g__zebra_listbody_#1_tl } }
\tl_new:N \g__zebra_listbody_all_tl
\cs_new:Npn \__zebra_use_listbody:n #1
  { \tl_use:c { g__zebra_listbody_#1_tl } }
\cs_new_protected:Npn \__zebra_record_note:nnnnn #1#2#3#4#5
  {
    \tl_gput_right:cx
      {
        \bool_if:NTF \l__zebra_sort_none_bool
          { g__zebra_listbody_all_tl }
          { g__zebra_listbody_#1_tl }
      }
      {
        \exp_not:N \__zebra_list_entry:nnnnn
```

```
        { \exp_not:n {#1} }
        {#2}
        { \exp_not:n {#3} }
        { \exp_not:n {#4} }
        {#5}
      }
  }
%% \__zebra_note_unnumbered:nnn {type}{author}{body}
%% Fast path for the \opt{unnumbered} option: skip dedup, .aux
%% signatures and list registration entirely.  The counter is still
%% bumped on every call (so the summary table reflects rough counts;
%% notes inside captions/TOC/marks may be over-counted, which is the
%% accepted trade-off for cutting out the fragile bookkeeping).
\cs_new_protected:Npn \__zebra_note_unnumbered:nnn #1#2#3
  {
    \bool_if:NT \l__zebra_draft_bool
      {
        \tl_set:Nx \l__zebra_note_color_tl { \__zebra_note_color:n {#1} }
        \int_gincr:c { g__zebra_note_count_#1_int }
        \__zebra_render_note:nnn {#1} {#2} {#3}
      }
  }
%% \__zebra_note:nnn {type}{author}{body}
%% Main entry point: dispatch to the lightweight unnumbered path or to
%% the numbered path with deduplication/bookkeeping.
\cs_new_protected:Npn \__zebra_note:nnn #1#2#3
  {
    \bool_if:NTF \l__zebra_unnumbered_bool
      { \__zebra_note_unnumbered:nnn {#1} {#2} {#3} }
      { \__zebra_note_numbered:nnn {#1} {#2} {#3} }
  }
%% Numbered path.  Four cases:
%%   Case 1 - sbox reuse:   instance_key found → reuse, render
%%   Case 2 - stable key:   label found, stable_key in prop → suppress
%%   Case 3 - content sig:  sig in .aux data → suppress
%%   Case 4 - new note:     allocate, render
\cs_new_protected:Npn \__zebra_note_numbered:nnn #1#2#3
  {
    \bool_if:NT \l__zebra_draft_bool
      {
        \tl_set:Nx \l__zebra_note_color_tl { \__zebra_note_color:n {#1} }
        %% Case 1: sbox reuse (same \inputlineno)
        \tl_set:Nx \l__zebra_note_key_tl
          { \__zebra_instance_key:nnn {#1} {#2} {#3} }
        \prop_get:NVNTF \g__zebra_note_target_prop \l__zebra_note_key_tl
            \l__zebra_note_target_tl
          {
            \prop_get:NVN \g__zebra_note_display_prop \l__zebra_note_key_tl
              \l__zebra_note_display_tl
            \__zebra_render_note:nnn {#1} {#2} {#3}
          }
          {
            %% Extract label and compute content signature
            \__zebra_extract_label:nN {#3} \l_tmpb_tl
```

```
\__zebra_content_sig:nnnN {#1} {#2} {#3} \l_tmpa_tl
%% Case 2: stable-key suppress (labeled, marks/headers).
%% If the stable key exists AND the content signature
%% matches, this encounter is a replay → suppress.
%% Different content signature = label reuse → warn and
%% let Case 4 allocate independently.
\bool_set_false:N \l_tmpa_bool
\tl_if_empty:NF \l_tmpb_tl
  {
    \tl_set:Nx \l__zebra_note_key_tl
      { \__zebra_stable_key:nn {#1} { \l_tmpb_tl } }
    \prop_get:NVNT \g__zebra_note_stable_sig_prop
        \l__zebra_note_key_tl \l_tmpc_tl
      {
        \tl_if_eq:NNTF \l_tmpa_tl \l_tmpc_tl
          { \bool_set_true:N \l_tmpa_bool }
          {
            \msg_warning:nnV { zebra }
              { duplicate-note-label } \l_tmpb_tl
          }
      }
  }
%% Case 3: content-sig suppress (labeled, TOC replay).
%% Only suppress if the stored instance key differs from
%% the current one - same key means it is the original
%% note, not a replay.
\bool_if:NF \l_tmpa_bool
  {
    \prop_get:NVNT \g__zebra_note_sig_known_prop
        \l_tmpa_tl \l_tmpc_tl
      {
        \tl_set:Nx \l_tmpd_tl
          { \__zebra_instance_key:nnn {#1} {#2} {#3} }
        \tl_if_eq:NNF \l_tmpc_tl \l_tmpd_tl
          { \bool_set_true:N \l_tmpa_bool }
      }
  }
\bool_if:NF \l_tmpa_bool
  {
    %% Case 4: new note - allocate and render
    \tl_set:Nx \l__zebra_note_key_tl
      { \__zebra_instance_key:nnn {#1} {#2} {#3} }
    \__zebra_allocate_note:nnn {#1} {#2} {#3}
    \prop_gput:NVV \g__zebra_note_target_prop
      \l__zebra_note_key_tl \l__zebra_note_target_tl
    \prop_gput:NVV \g__zebra_note_display_prop
      \l__zebra_note_key_tl \l__zebra_note_display_tl
    %% For labeled notes: register stable key + write sig,
    %% but only if the stable key is not already claimed by
    %% an earlier note (label-conflict case).
    \tl_if_empty:NF \l_tmpb_tl
      {
        \tl_set:Nx \l__zebra_note_key_tl
          { \__zebra_stable_key:nn {#1} { \l_tmpb_tl } }
```

```
                    \prop_if_in:NVF \g__zebra_note_stable_sig_prop
                        \l__zebra_note_key_tl
                      {
                        \prop_gput:NVV \g__zebra_note_target_prop
                          \l__zebra_note_key_tl \l__zebra_note_target_tl
                        \prop_gput:NVV \g__zebra_note_display_prop
                          \l__zebra_note_key_tl \l__zebra_note_display_tl
                        \prop_gput:NVV \g__zebra_note_stable_sig_prop
                          \l__zebra_note_key_tl \l_tmpa_tl
                        \tl_set:Nx \l_tmpc_tl
                          { \__zebra_instance_key:nnn {#1} {#2} {#3} }
                        \__zebra_write_sig:NN \l_tmpa_tl \l_tmpc_tl
                      }
                  }
                \__zebra_render_note:nnn {#1} {#2} {#3}
              }
            %% Cases 2-3: suppress - no output
          }
      }
  }
%% Full render: hypertarget, target label, margin note, inline text.
%% In \opt{unnumbered} mode the label/hypertarget setup is skipped so a
%% \cs{label} inside the note body falls through to the surrounding
%% \cs{@currentlabel} (e.g.\ the section number), and \cs{zebraref}
%% degrades cleanly to \cs{ref}.
\cs_new_protected:Npn \__zebra_render_note:nnn #1#2#3
  {
    \group_begin:
      \bool_if:NF \l__zebra_unnumbered_bool
        {
          \protected@edef \@currentlabel { \l__zebra_note_display_tl }
          \__zebra_if_package_loaded:nT { hyperref }
            { \tl_set:Nx \@currentHref { \l__zebra_note_target_tl } }
          \tl_set:Nx \l__zebra_note_ref_type_tl
            { \__zebra_capitalize_type:n {#1} }
          \__zebra_target:nn { \l__zebra_note_target_tl } {}
          \exp_args:NV \__zebra_kernel_label:n \l__zebra_note_target_tl
        }
      \__zebra_margin_note:nn
        {\textcolor{\l__zebra_note_color_tl}{%
          \bool_if:NF \l__zebra_unnumbered_bool
            { {\bfseries\l__zebra_note_display_tl}\kern1pt }%
          \__zebra_note_symbol:n {#1}}}
        {\textcolor{\l__zebra_note_color_tl}{%
          \__zebra_note_symbol:n {#1}%
          \bool_if:NF \l__zebra_unnumbered_bool
            { \kern1pt {\bfseries\l__zebra_note_display_tl} }}}%
      \bool_if:NF \l__zebra_unnumbered_bool
        { \cs_set_eq:NN \label \__zebra_note_label:n }
      \textcolor{\l__zebra_note_color_tl}{[\colorbox[gray]{0.97}{%
          \textcolor{\l__zebra_note_color_tl !70!black}{%
            \textsc{\MakeLowercase{\MakeUppercase#1}}%
            \bool_if:NF \l__zebra_unnumbered_bool
              { ~\l__zebra_note_display_tl }%
```

```
            \texttt{\__zebra_prepend:nn {@}{#2}:}} #3]}%
    \group_end:
  }
\cs_new_protected:Npn \__zebra_new_note_type:nn #1#2
  { \__zebra_new_note_type:nnn {#1} {#2} {} }
\cs_new_protected:Npn \__zebra_new_note_type:nnn #1#2#3
  {
    \seq_gput_right:Nn \g__zebra_note_types_seq {#1}
    \prop_if_in:NnF \g__zebra_note_colors_prop {#1}
      { \prop_gput:Nnn \g__zebra_note_colors_prop {#1} {#2} }
    \tl_if_blank:nF {#3}
      {
        \prop_if_in:NnF \g__zebra_note_symbols_prop {#1}
          { \prop_gput:Nnn \g__zebra_note_symbols_prop {#1} {#3} }
      }
    \int_new:c { g__zebra_note_count_#1_int }
    \__zebra_new_listbody:n {#1}
    \exp_args:Nc \NewDocumentCommand { zebra#1 } { O{} m }
      { \__zebra_note:nnn {#1}{##1}{##2} }
    \__zebra_if_package_loaded:nT { hyperref }
      { \__zebra_apply_pdfstring_defs: }
    \cs_if_exist:cTF {#1}
      { \msg_warning:nnn { zebra } { command-taken } {#1} }
      {
        \cs_set_eq:cc {#1} {zebra#1}
        \prop_gput:Nnn \g__zebra_note_public_alias_prop {#1} { true }
      }
  }
\cs_new_protected:Npn \__zebra_list_entry:nnnnn #1#2#3#4#5
  {
    \par\noindent
    \textcolor{\__zebra_note_color:n {#1}}{%
      \textbf{\__zebra_capitalize_type:n {#1}~#2}%
      \tl_if_blank:nF {#3} { \enspace \texttt{\__zebra_prepend:nn {@}{#3}} }}%
    \nobreak\dotfill
    \__zebra_link:nn {#5} { \__zebra_pageref:n {#5} }%
    \par
    \begingroup
      \leftskip=2em
      \rightskip=2em
      \parindent=0pt
      \cs_set_eq:NN \label \use_none:n
      #4\par
    \endgroup
  }
\cs_new_protected:Npn \__zebra_print_note_group:n #1
  {
    \int_compare:nNnT { \__zebra_note_count:n {#1} } > { 0 }
      {
        \par\medskip
        \__zebra_use_listbody:n {#1}
      }
  }
\cs_new_protected:Npn \__zebra_print_notes_inorder:
```

```
  {
    \tl_if_empty:NF \g__zebra_listbody_all_tl
      { \par\medskip \tl_use:N \g__zebra_listbody_all_tl }
  }
\cs_new_protected:Npn \__zebra_summary_row:n #1
  {
    \int_compare:nNnT { \__zebra_note_count:n {#1} } > { 0 }
      {
        \int_add:Nn \l__zebra_total_notes_int { \__zebra_note_count:n {#1} }
        \tl_put_right:Nx \l__zebra_summary_rows_tl
          {
            \exp_not:N \textcolor
              { \__zebra_note_color:n {#1} }
              {#1}
            \exp_not:N &
            \__zebra_note_count:n {#1}
            \exp_not:N \\
          }
      }
  }
\cs_new_protected:Npn \__zebra_print_notes:
  {
    \tl_clear:N \l__zebra_summary_rows_tl
    \int_zero:N \l__zebra_total_notes_int
    \seq_map_inline:Nn \g__zebra_note_types_seq
      { \__zebra_summary_row:n {##1} }
    \tl_if_empty:NF \l__zebra_summary_rows_tl
      {
        \par\nobreak
        \noindent\dotfill\par\medskip
        \nobreak
        \noindent\textbf{\Large Zebra~Notes}
        \par \medskip
        \begin{center}
          \begin{tabular}{lr}
            \hline
            \textbf{Type} & \textbf{Count} \\\hline
            \tl_use:N \l__zebra_summary_rows_tl
            \hline
            \textbf{Total} & \textbf{\int_use:N \l__zebra_total_notes_int} \\
            \hline
          \end{tabular}
        \end{center}
        \bool_if:NF \l__zebra_unnumbered_bool
          {
            \par \medskip
            \begingroup
              \small
              \noindent{\bfseries List~of~notes}\par
              \nobreak
              \bool_if:NTF \l__zebra_sort_none_bool
                { \__zebra_print_notes_inorder: }
                {
                  \seq_map_inline:Nn \g__zebra_note_types_seq
```

```
                    { \__zebra_print_note_group:n {##1} }
                }
              \endgroup
            }
        }
    }
```

\zebranewnote All note types are created with \zebranewnote.

```
\NewDocumentCommand \zebranewnote { m m O{} }
  { \__zebra_new_note_type:nnn {#1} {#2} {#3} }
```

(*End of definition for* \zebranewnote. *This function is documented on page* 4.)

\zebrasetup Applies configuration keys after loading using the zebra-setup key family.

```
\NewDocumentCommand \zebrasetup { m }
  { \keys_set:nn { zebra-setup } {#1} }
```

(*End of definition for* \zebrasetup. *This function is documented on page* 5.)

\todo \note \fixed \comment \placeholder Built-in note types, defined with \zebranewnote.

```
\zebranewnote{todo}{purple}
\colorlet{zebra@fixed@color}{green!50!black}
\zebranewnote{fixed}{zebra@fixed@color}
\zebranewnote{comment}{blue}
\zebranewnote{note}{violet}
\zebranewnote{placeholder}{gray}
```

(*End of definition for* \todo *and others. These functions are documented on page* 3.)

4.4 Print summary at end of the document

A summary table and a detailed note list are inserted automatically at the end of the document. Each note type with at least one instance is listed with its colour and count, followed by notes in document order or grouped by type.

```
%% At end of document: print the note summary and list.
%% Content signatures are written to .aux inline (at allocation time),
%% so no additional end-of-document aux writes are needed.
\hook_gput_code:nnn { enddocument } { zebra }
  {
    \bool_if:NT \l__zebra_draft_bool
      { \__zebra_print_notes: }
  }
\ExplSyntaxOff
```

⟨/package⟩

4.5 Compatibility shim

The old package name zebra-goodies is supported via a thin wrapper that loads zebra and prints a deprecation warning.

```
⟨*compat⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{zebra-goodies}
  [2026/04/26 v1.8.0 Deprecated: use zebra instead]
\PackageWarningNoLine{zebra-goodies}
  {Package 'zebra-goodies' is deprecated.\MessageBreak
   Use \string\usepackage{zebra} instead}
\RequirePackageWithOptions{zebra}
⟨/compat⟩
```

4.6 Two-column demo

A standalone two-column document used to generate the demo figure included in the documentation. It is extracted automatically by docstrip and compiled during the build.

```
⟨*demo-twocol⟩
\documentclass[twocolumn]{article}
\usepackage[paperwidth=21cm,paperheight=15cm,margin=1.1cm]{geometry}
\usepackage{zebra}
\zebrasetup{symbol/comment=$\clubsuit$}
\pagestyle{empty}
\begin{document}
\section{Demo name\comment{revise the name}}
This draft still needs work
\todo[alice]{\label{zebra:intro}revise the introduction}. The
opening paragraph should also explain the main goal more plainly.
Add one more citation here \note[bob]{support this claim}. A
brief roadmap sentence would also make the structure easier to
scan.

The issue raised in Todo~\ref{zebra:intro} still applies in
the conclusion. The table now looks fine
\fixed[carol]{alignment corrected}, but one figure is still
missing \placeholder[eve]{insert the overview figure}. A short
transition would also help the flow. The middle section should
probably end with a clearer summary sentence before the
discussion begins. That summary can stay compact, but it should
signal why the next section matters.

Please verify the totals \note[frank]{check the numbers} and
confirm the wording in the last paragraph
\comment[tom]{is this sentence too strong?}. A small typo has
already been fixed \fixed[heidi]{typo corrected}. The ending
should stay short. The final sentence should return to the main
claim rather than repeat background material. You can place
\todo[judy]{summarise the findings} anywhere once the narrative
is stable.

One more short paragraph is enough to show how \placeholder{wow,
so great!} the markers stay readable in a compact two-column layout.
```

```
669 The example is intentionally small, but it should still look like
670 a realistic revision pass.\comment{Bye}
671 \end{document}
672 ⟨/demo-twocol⟩
```

Change History

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

U

V

W

Z

Zebra Notes

Type	Count
todo	4
fixed	1
comment	1
note	3
placeholder	1
question	1
Total	**11**

List of notes