

Table of Contents

1. Website macros	2
1.1 Quick start	2
1.2 Variables	2
1.2.1 Why <code>pdk_repo_url</code> and <code>pdk_applied_release</code> remain Prodockit variables	3
1.3 Macros	3
1.4 Show macro syntax as text	4
1.4.1 <code>pdk_heading_counter_reset(page)</code>	4
1.4.2 <code>pdk_reference_style()</code> / <code>pdk_acronym_style()</code> / <code>pdk_glossary_style()</code>	4
1.5 Page dates are not macros	5

1. Website macros

`prodockit.zensical_macros` provides a handful of Jinja variables and macros for Zensical's own [macros plugin](#) - the pieces a professional/academic report's website commonly wants that aren't specific to any one project: a site-wide word count, the git-detected repository URL, the successfully applied template release, chapter/appendix numbering that continues across pages, and reference/acronym/glossary list spacing that matches [prodockit.pdf](#)'s own PDF output.

1.1 Quick start

Add it alongside your own project's `macros.py` (which keeps anything genuinely project-specific - a custom macro, institution branding, and so on):

```
[project.markdown_extensions.zensical.extensions.macros]
module_name = "macros"
modules = ["prodockit.zensical_macros"]
on_error_fail = true
```

Zensical's macros plugin loads `module_name` and every entry in `modules`, merging all of their variables/macros into the same Jinja environment - so a project with no macros of its own can drop `module_name / macros.py` entirely and just use:

```
[project.markdown_extensions.zensical.extensions.macros]
modules = ["prodockit.zensical_macros"]
on_error_fail = true
```

Keep `on_error_fail = true` in automated and local builds. If Jinja cannot render a page variable or macro call, Zensical then stops the build instead of returning the unrendered page and allowing a broken site to be published.

1.2 Variables

Zensical already exposes project configuration through `config` and repository metadata through `git`. Prodockit adds only values with different semantics. Use the `pdk_` prefix for its public names to avoid collisions with project and plugin names. The old unprefixed names remain temporarily for compatibility; old macro calls emit a deprecation warning. The values used by Prodockit projects are listed in [Table 29.1](#).

Table 1.1. Variables

Variable	Description
<code>{{ pdk_word_count }}</code>	Prose word count across every nav page except the first (assumed to be the cover page) and any page flagged <code>exclude_from_word_count: true</code> in its own front matter - a comma-formatted string (e.g. "9,971").

Variable	Description
<code>{{ pdk_repo_url }}</code>	The fully-qualified <code>https://</code> URL for the current checkout's git origin remote (converted from <code>git@host:path.git</code> SSH syntax, with any embedded CI credentials stripped) - <code>""</code> if there's no git remote configured.
<code>{{ pdk_applied_release }}</code>	The <code>prodockit-template</code> release most recently applied successfully. Bootstrap initialises it from the highest versioned release tag reachable in the pristine template's history; <code>template-sync --apply</code> then updates the persisted <code>.prodockit-template</code> value only after applying a template update. A student's own repository tags cannot change it.
<code>{{ config.site_name }}</code>	Native Zensical value for <code>project.site_name</code> from <code>zensical.toml</code> . Prefer it to the removed Prodockit <code>site_name</code> alias.
<code>{{ git.short_tag }}</code>	Native Zensical value for the nearest reachable tag in the current documentation repository. Prefer it to the removed Prodockit <code>release</code> alias when showing the document's own release. This is deliberately different from <code>pdk_applied_release</code> .

1.2.1 Why `pdk_repo_url` and `pdk_applied_release` remain Prodockit variables

Zensical's built-in macro context already exposes the project configuration through `config` and repository metadata through `git`. Prodockit uses native values such as `config.site_name` and `git.short_tag` where their semantics match. The two variables below deliberately add behaviour that those values do not currently provide. See [Zensical's built-in template variables](#) for the native interface they complement.

`{{ pdk_repo_url }}` describes the checkout being built, not only the URL last written to `zensical.toml`. It reads the active `origin`, converts Git's SSH form to an ordinary HTTPS link, and removes embedded CI credentials before the value reaches generated HTML. This keeps a fork or mirror pointing at its real repository and prevents a token-bearing clone URL from becoming a public link. The native `config.repo_url` remains the right choice when the configured value is intentionally different from the active checkout.

`{{ pdk_applied_release }}` records the version of `prodockit-template` most recently applied successfully, not the latest tag in the student's repository. The native `git.short_tag` is therefore the right value for the document's own release but cannot describe its template state. Keeping these meanings separate lets maintainers see whether a project has successfully received a template fix even after the project creates its own tags.

The prefixed Prodockit names are stable author-facing interfaces. If a future Zensical release provides the same normalized, credential-safe repository URL or an equivalent persisted template-release value, Prodockit will implement the matching variable as a compatibility alias to Zensical's native value. Authors will not need to rewrite existing `{{ pdk_repo_url }}` or `{{ pdk_applied_release }}` expressions, while Prodockit can stop maintaining duplicate discovery logic.

1.3 Macros

[Table 29.2](#) lists the callable helpers and the content each one inserts.

Table 1.2. Macros

Macro	Description
<code>{{ pdk_heading_counter_reset(page) }}</code>	Place near the top of every page - continues chapter/section numbering (and the matching sidebar numbering) across pages, from this page's position in nav. See below.
<code>{{ pdk_reference_style() }}</code>	Place once near the top of a references page - controls <code>.reference</code> paragraph spacing. See below.
<code>{{ pdk_acronym_style() }}</code>	Place once near the top of an acronyms page - matches <code>pdk_reference_style()</code> 's default spacing.
<code>{{ pdk_glossary_style() }}</code>	Place once near the top of a glossary page - matches <code>pdk_reference_style()</code> 's default spacing.

1.4 Show macro syntax as text

The macros plugin processes Jinja delimiters before Markdown code formatting. Backticks therefore do not protect a literal macro example. A literal `{{ pdk_word_count }}`, GitHub expression, or compact `{#heading-id}` example can stop every macro on that page from being rendered.

When readers should see the syntax rather than run it, wrap the literal text between `{% raw %}` and `{% endraw %}` in the Markdown source. For example:

```
{% raw %}
{{ word_count }}
${{ github.token }}
{#heading-id}
{% endraw %}
```

The raw wrapper is removed from the website and the intended braces remain. Use an unwrapped expression when it is meant to run.

1.4.1 `pdk_heading_counter_reset(page)`

Continues heading numbering from wherever the previous page left off. The numbering stays aligned with `\ref{}` links and updates when pages are reordered or headings are added or removed.

Set `project.extra.heading_numbering = false` in `zensical.toml` to turn numbering off entirely (content and sidebar) across the whole site. A page flagged `is_appendix: true` in its own front matter gets letter-based numbering instead - "Appendix A", "A.1", "A.1.1" - matching `prodockit.headings`' own `appendix_attr` default.

Contributors changing how page numbers are discovered should read [Extension integration](#).

1.4.2 `pdk_reference_style()` / `pdk_acronym_style()` / `pdk_glossary_style()`

Controls list-entry spacing, driven by the same `project.extra.*` settings [prodockit.pdf](#) reads for the PDF, so both outputs stay in sync from one configured value:

[Table 29.3](#) shows which configured style value each helper returns.

Table 1.3. `pdk_reference_style()` / `pdk_acronym_style()` / `pdk_glossary_style()`

Setting	Default	What it does
<code>reference_style</code>	"european"	"european" : single line spacing throughout, no indent, entries close together. "global" : single line spacing within each entry, double spacing <i>between</i> entries, with a hanging indent on wrapped lines (the common APA/MLA/Chicago style). Only <code>pdk_reference_style()</code> / the References page switches look - acronyms/glossary always use the tight "european" spacing.
<code>reference_spacing_european</code>	"-0.8em"	Gap between entries, "european" style - also used unconditionally for the acronym/glossary lists.
<code>reference_indent_global</code>	"1.27cm"	Hanging indent on wrapped lines, "global" style.
<code>reference_spacing_global</code>	"2em"	Gap between entries, "global" style.

For supported versions and the pre-1.0 stability boundary, see [Support and compatibility](#).

1.5 Page dates are not macros

Build-derived dates do not require the macros plugin. Use [Page update dates](#) to choose where a website date appears, write its accompanying text, or override one page's automatic date.