Table of Contents

1. Release Notes

1.1 0.10.7 (2026-07-25)

Two numbering fixes, both cases of the same shape: a raw-text pre-scan and a parsed-document count applying different rules to the same content.

Website and PDF disagreeing on appendix letters. Appendix lettering was computed twice. The website gave a letter to any page whose front matter set `is_appendix`, unconditionally; the PDF's Lua filter counted appendix h1s instead. An appendix page contributing no numbered h1 - none at all, or one marked `unnumbered`, which the filter skips - gave Lua nothing to count, so every later appendix came out a letter early. Reproduced against `prodockit-template`: the same Bibliography page rendered as "Appendix E" on the website but "Appendix D" in the PDF.

`prodockit.pdf.build` now assigns every appendix page's letter once, by position in the page list, and stamps it on that page's heading for `Header()` to read rather than counting for itself. A page with no numbered h1 still consumes its letter, so the two stay in step.

Setext headings invisible to the nav pre-scan. `_count_top_level_headings()` matched ATX headings only, so a title underlined with `=` was never counted - though Zensical's renderer and Pandoc both produce a real h1 either way, and both number it. Later pages' start counts came out short, which broke numbering twice over: the website contradicted itself, giving the next page the chapter number a setext heading had already taken, and it fell one behind the PDF. Each rule of the setext syntax was checked against the real renderer rather than assumed - a single `=` is enough, a `-` underline is an h2, a two-line paragraph followed by `=====` is no heading at all, and `attr_list` puts `{: .unnumbered }` on the text line.

Stale cross-page references under `zensical serve`. `preseed()` is deliberately first-wins so a duplicate id resolves to the first page in nav order. Under `zensical build` that is all it has to do; under `zensical serve` the process outlives the files, and first-wins silently discarded fresh data - an edited definition kept its original text, and a deleted one stayed resolvable, until the dev server restarted. `preseed_attr_from_nav()` now rebuilds the provisional set from scratch on each scan, and `prodockit.headings` keys its cached scan on every nav page's mtime and size as well as the numbering settings, so an edit to a page a given render doesn't touch still invalidates it.

Note this fixes the pre-scan, not Zensical's incremental rebuild: verified against a live `zensical serve`, editing page A does not cause page B to re-render, so B's output still only catches up when B itself is re-rendered. What is guaranteed now is

that a page being re-rendered resolves against current disk state rather than a snapshot from server startup.

Fixes #104, #106 and #99.

1.2 0.10.6 (2026-07-25)

Fixed footnote text in the PDF rendering in a column roughly two thirds of the page's content width, wrapping a footnote onto five short lines whose first held just two words.

This had been documented in `prodockit.pdf.css`'s own `.pdf-footnote` rule as an unfixable WeasyPrint `float: footnote` limitation being tracked upstream. That was a misdiagnosis. The real cause is Pandoc's HTML writer hard-wrapping its output at ~72 columns, inserting newlines *inside* the `<span class="pdf-footnote">` carrying a footnote's text. Those newlines are insignificant whitespace in HTML, but WeasyPrint's `float: footnote` width computation treats them as hard break opportunities when sizing the footnote area, so the rendered text collapses toward the longest *source* line rather than the page's content width.

Confirmed by holding the HTML and CSS constant and varying only the Pandoc step: the same document rendered 304.1pt wide through Pandoc but 462.9pt straight through WeasyPrint. `prodockit.pdf.build` now passes `--wrap=none`, giving 474.2pt of a ~482pt content width. WeasyPrint 69.0 is already the latest release, so waiting upstream had no path forward.

This does not stop footnotes wrapping in the PDF - `--wrap=none` governs only newlines in the generated HTML source, never the engine's own line breaking. A long footnote still occupies as many lines as it needs, each now using the full measure: a seven-sentence footnote renders on six full-width lines rather than ten narrow ones.

The misleading `KNOWN LIMITATION` comment in `prodockit.pdf.css` has been replaced with the real cause so it isn't re-derived, and two regression tests added - one asserting the flag at the command level (so it can't be dropped where Pandoc isn't installed), one measuring real rendered text width via a genuine Pandoc/WeasyPrint build, since the CSS is identical either way and only a real render can tell the two apart.

Fixes #101, reported downstream as prodockit-template#95.

1.3 0.10.5 (2026-07-25)

Fixed a real bug found by reproducing it directly: a cross-page `\ref{id}` under `zensical build` depended on whether the page defining `id` happened to be rendered before the page referencing it, in the same Python process - `zensical build` renders pages neither in nav order nor necessarily all in one process, so

this was pure luck. Reproduced locally on a 3-page site: the previous release left 1-5 references as `??` per build, varying from one otherwise-identical build to the next (only 2 of 12 clean builds fully resolved).

`prodockit.headings` now pre-scans every nav page's headings (ids, levels, section numbers) into the shared registry before any page is converted - the same idea `prodockit.citations` / `prodockit.glossary` already use for their own cross-page definitions - so resolution no longer depends on build order. A page actually rendered in this process still supersedes its own pre-scanned entries with the real ones. 20 consecutive clean builds now produce byte-identical, fully-resolved output; `prodockit-template`'s entire built site is byte-identical before and after this change.

Also fixed a second bug found while testing the above: extension order isn't guaranteed, so `prodockit.refs` can construct its own default `HeadingsExtension` and trigger the pre-scan with per-document numbering before a project's configured `numbering = "continuous"` instance runs - silently showing a cross-page reference's number one step behind (`1.1` instead of `2.1`) roughly 1 build in 12. The pre-scan now reruns if a differently-configured instance appears.

Fixes #54. See #99 for a related, separate limitation found along the way (this pre-scan can go stale under `zensical serve`'s live-reload - not fixed here).

1.4 0.10.4 (2026-07-25)

- Added `CONTRIBUTING.md` and a `.github/pull_request_template.md`, adapted from prodockit-template's own - library-specific setup (`pip install -e ".[dev]"`, `pytest` / `ruff` / `mypy`, the real-`pandoc` requirement for `prodockit.bibliography`'s own tests) rather than the template's assignment-writing framing. Linked from README.md's Development section.
- Docs: added an admonition to `headings.md` documenting a real gap this project's own docs hit directly while enabling every prodockit extension on its own site (#87) - Zensical's automatic cross-page id sharing only warns (rather than raising) on a heading name shared across pages, and build order isn't guaranteed stable, so the "keeping the first" winner can change between builds. Documents the fix - an explicit, page-prefixed id via `attr_list` - pointing to this project's own docs as a worked example.

No code changes.

1.5 0.10.3 (2026-07-25)

Docs: several extension pages mixed "why it was built this way" design rationale into their opening paragraph, ahead of the practical "what it does"/"how to use it"

content most readers want first - moved that reasoning further down each page instead:

- `bibliography.md`: trimmed the intro to the core value proposition, moved the Pandoc-delegation rationale and architecture diagram out of Requirements (now just what to install) into a new "How it works" section after Reference, and folded the "can be enabled alongside prodockit.citations" note into Comparing the two approaches.
- `citations.md` / `glossary.md`: moved the "why bundled into one extension, unlike headings/refs" rationale from the intro into their own Syntax section.
- `tables.md`: moved the "auto-enables Python-Markdown's own tables extension" implementation note from the intro into Syntax.
- `index-terms.md`: moved the "why a Markdown extension, not attr_list" rationale from the intro to the end of CSS hooks, wrapped in its own admonition naming the subject explicitly.

`headings.md` / `refs.md` were already lean and needed no changes. No syntax, behaviour, or code changes.

1.6 0.10.2 (2026-07-25)

Docs: `extensions/index-terms.md` described the live website's search as generic "browser/Ctrl-F search" - updated to point at Zensical's own built-in site search instead, a more accurate and discoverable description of how a reader actually finds a term on the live website. No code changes.

1.7 0.10.1 (2026-07-24)

Docs: `prodockit.index`'s marking syntax and `prodockit.pdf`'s back-of-book index *generation* were split across two pages (`extensions/index-terms.md` and `pdf.md` respectively), even though the marker is useless without turning `pdf_include_index` on and vice versa - `prodockit.bibliography`'s own docs already combine marking and generation into one page. Moved the generation content into `extensions/index-terms.md` as a new "Generating the index" section, merged the per-feature rendered-output examples into their existing marking sections instead of duplicating them, and renamed the page from "Index terms" to "Index (pdf-only)" now that it covers the whole feature. `pdf.md` keeps only a short pointer, matching how `prodockit.bibliography` is treated there. No code changes.

1.8 0.10.0 (2026-07-24)

`prodockit.bibliography`'s `\bibliography` marker now takes two optional, positional parameters - `\bibliography{<file>}{<true|false>}` - so a project can generate both a strict **References** section (only sources actually `\citebib{}`-cited in the text) and a broader **Bibliography** section (every entry,

including background reading that's never individually cited) in one build, from the same or different `.bib` files:

```
<!-- references.md -->
\bibliography{}{true}
```

```
<!-- bibliography.md -->
\bibliography{background.bib}
```

Bare `\bibliography` is completely unchanged - fully backward compatible, no breaking change. A `\citebib{id}` citation now cross-links to whichever marker's page actually defines that entry (via a new, lightweight `.bib` entry-key discovery helper, not a CSL reimplementation) rather than assuming a single global bibliography page - the common single-file case is unaffected. See Multiple sections: References and Bibliography for the full syntax and worked examples.

Fixes #89.

1.9 0.9.0 (2026-07-24)

Breaking: `copyright` / `pdf_copyright` are now a real HTML fragment, rendered as a real DOM element in the PDF's running footer via CSS Paged Media's `position: running()` / `content: element()`, instead of being escaped into a CSS `content: "..."` string. This is what makes a real `<a href="../...">` link inside either value survive as a real, clickable link in the PDF - on every page, not just wherever the source element itself sits - matching how Zensical's own website-side `copyright` setting already works. Use a real `<br>` for a forced line break; the `\A` CSS-escape trick added in 0.8.0 only ever worked for a plain string, not real markup, and no longer applies - update any existing `pdf_copyright` using it to a real `<br>` instead.

`prodockit.pdf.css.build_css()` no longer takes a `copyright_text` parameter (`site_name` is unaffected, still a plain CSS content string) - no formal, versioned public API surface yet for `prodockit.pdf` (see prodockit-extension#7), so this is an acceptable break at this stage.

This project's own cover page (`docs/index.md`'s hero subtitle) no longer hyperlinks the word "Zensical" - it stays as plain text, matching this project's own PDF footer now crediting Zensical/prodockit with real links instead of the cover page doing it via a website-only, PDF-invisible link.

1.10 0.8.1 (2026-07-24)

Docs: this project's own docs site and PDF were missing the "Made with Zensical and prodockit" credit line that `overrides/partials/ copyright.html` / `pdf_copyright` (new in 0.8.0) already give a downstream project - added both

here too, via a new `overrides/partials/ copyright.html` for the website and `extra.pdf_copyright` in `zensical.toml` for the PDF, so this site credits itself the same way a project built with it does. No library code changed.

1.11 0.8.0 (2026-07-24)

New `pdf_copyright` setting: `project.copyright` (a plain, native Zensical setting) already feeds the footer of both the website and the PDF by default - `pdf_copyright` is an opt-in override for the PDF's footer only, for a project that wants its PDF footer to say something different from its website's (e.g. adding a "Made with Zensical and prodockit" credit line only to the downloadable PDF, not the live site). Write a forced line break in either setting with a literal `\A` inside a TOML *literal* string (`'''...'''`) - see Copyright text for the full mechanism and why a literal string is required.

Also fixed a real, previously-undocumented rendering gap found while building this: a `\A` forced line break inside a `content` string only actually renders as a line break under `white-space: pre-line` - under WeasyPrint's default `white-space: normal` it silently collapsed to a plain space instead. Both the single-sided and double-sided verso copyright footer boxes now set `white-space: pre-line` so the forced break always works as expected.

1.12 0.7.1 (2026-07-24)

This project's own documentation site now enables every prodockit extension (`prodockit.headings`, `prodockit.refs`, `prodockit.citations`, `prodockit.glossary`, `prodockit.bibliography`, in addition to the `prodockit.tables` / `prodockit.index` already enabled) via `zensical.toml`, dogfooding the full set rather than just the two used to build this site previously.

Doing so surfaced a real bug: Zensical does not render pages in a stable order between builds, so a heading name shared across two or more pages (e.g. "Quick start", "Syntax", "Options") non-deterministically resolves its id collision differently from one `zensical build` to the next - confirmed by running repeated clean builds and observing the reported "keeping the first" winner change between runs. Fixed by giving every colliding heading across the docs an explicit, unique, page-prefixed id via `attr_list` (e.g. `## Quick start {: #refs-quick-start }`), rather than relying on build order at all. No library code changed - this is a docs-content-only fix, and not something a project sharing a heading name across its own pages will normally have to think about, since a one-off name collision is far less likely there than in this project's consciously-parallel per-extension documentation structure.

1.13 0.7.0 (2026-07-24)

Breaking: `prodockit.bibliography` now uses its own `\citebib{id}` syntax instead of `\cite{id}`. Previously it registered the same `\cite{id}` pattern

`prodockit.citations` uses, at the same inline-pattern priority - enabling both extensions together left it undefined which one actually resolved a given `\cite{...}` occurrence. Renaming `prodockit.bibliography`'s own syntax removes the conflict entirely: both extensions can now be enabled in the same build with no interference, each citing its own sources by its own marker. A project still using `prodockit.bibliography` on its own needs to update every `\cite{id}` in its source to `\citebib{id}` - the old syntax no longer resolves.

1.14 0.6.8 (2026-07-21)

`build_pdf_from_zensical_config()` (what `prodockit pdf` runs) now supports cover page markers, so a project no longer needs its own custom Python just to fill in a cover page's word count/repo URL/release tag - found via `prodockit-template`, whose `build_pdf.py` had grown to nearly nothing except this one piece:

- `{WORDCOUNT}` - the site-wide word count (the same value a `{{ word_count }}` website macro shows).
- `{REPOURL}` - the git-detected repo URL.
- `{RELEASE}` - the latest published GitHub/GitLab release tag - the whole line is dropped instead if there isn't one.
- `{{ site_name }}` - substituted literally, since `prodockit pdf` never evaluates Jinja.

All four are opt-in by literally writing the marker in your `nav`'s index page - no new `zensical.toml` setting needed. See Cover page markers.

Also new: `pdf_extra_css`, a stylesheet meant *only* for the PDF (e.g. a rule that would look wrong on the live website), concatenated after `extra_css` - the same `["stylesheets/print.css"]` role a project's own custom PDF-build script might have hardcoded outside `zensical.toml` entirely before, now expressible as ordinary configuration.

Also fixed two real bugs found while building this:

- `extra_css`'s (and now `pdf_extra_css`'s) own relative `url(...)` references (e.g. a light/dark logo swap or a header background image) were passed through unresolved, pointing nowhere once compiled into the PDF's own temporary work directory - now resolved and base64-embedded, matching how a local `<img>` reference already was.
- `copyright` / `site_name` were passed straight into `build_pdf()`'s generated CSS `content: "..."` string with no escaping at all - `project.copyright` is commonly a triple-quoted TOML string spanning multiple lines, and a raw embedded newline (or a literal `"`) silently broke the whole generated rule, dropping the running header/footer entirely with no error. Both are now collapsed to one line and escaped before being passed through.

1.15 0.6.7 (2026-07-21)

Fixed `prodockit.pdf.html.fix_up_page_html()` permanently embedding *both* halves of a `#only-light` / `#only-dark` (or GitHub's `#gh-light-mode-only` / `#gh-dark-mode-only`) image pair in a PDF, stacked one after the other, instead of just one - found via `prodockit-template`'s own cover page hero graphic showing twice. A PDF has no light/dark toggle to make that convention meaningful, but `to_base64_data_uri()` already strips anything from `#` onward before resolving the file (to find the right one), so the resulting `data:` URI has no trace of the fragment left for any stylesheet to hide either half by. The `#only-dark` / `#gh-dark-mode-only` half is now dropped entirely rather than embedded.

1.16 0.6.6 (2026-07-21)

- Docs: the cover page hero graphic (`docs/assets/cover-hero-*.svg`) used a different colour palette in light mode (blue) than in dark mode (green) - recoloured the light variant to match dark exactly, so the hero reads the same regardless of theme. The "Download PDF" button also picked up this same green, rather than the theme's default primary colour.
- `prodockit.pdf.css`'s back-of-book index letter-group headings (`h2.prodockit-index-letter` - the "A", "B", "C" separators) were hardcoded to the hero graphic's *old* light-theme blue - updated to match the now-green hero, which a PDF always shows regardless of a project's own website light/dark toggle.
- No functional (Python package behaviour) changes beyond the index letter colour.

1.17 0.6.5 (2026-07-21)

Extends the 0.6.4 always-excluded-directory mechanism in `prodockit.pdf.source_bundle` to two more classes of vendored, never student-written content:

- Any directory literally named `styles` - a Vale `StylesPath` (conventionally named this way) holds downloaded rule packs (e.g. the Microsoft, proselint, and Readability style guides), typically tracked for offline/CI builds rather than gitignored.
- Common dependency lockfiles by exact file name - `package-lock.json`, `npm-shrinkwrap.json`, `yarn.lock`, `pnpm-lock.yaml`, `Pipfile.lock`, `poetry.lock`, `Cargo.lock` - machine-generated by a package manager, never hand-written, and often thousands of lines each.

Neither is project-configurable, matching 0.6.4's `.icons` exclusion: a project can't reach for the same knob to narrow what a bundle archives.

Also fixes `source_bundle.pdf`'s running header naming the wrong file: a file's own last page could show the *next* file's name instead of its own, because the

invisible marker that sets the header text had no page break of its own - only the following content did - so it rendered on the tail end of the previous file's last page. The break now moves onto the marker itself, so the string-set and the page it applies to always agree.

1.18 0.6.4 (2026-07-21)

`prodockit.pdf.source_bundle` now always excludes any directory literally named `.icons` (e.g. a `custom_icons` directory, per `pymdownx.emoji`'s own convention) from `source_bundle.pdf`, regardless of `.gitignore` - found via `prodockit-template`, whose own vendored icon packs (`overrides/.icons/bootstrap`, `overrides/.icons/gitlab` - together ~2,500 unused SVGs) turned a source bundle meant to hold a student's own report content into a 3,000-page dump of unreferenced vendor assets. `.gitignore` alone can't fix this, since these directories are typically *tracked* (needed for the site/PDF to build at all) - deliberately not a project-configurable setting, so a project can't reach for the same knob to exclude something that should actually be archived.

1.19 0.6.3 (2026-07-20)

Bug fixes found by an in-depth test-coverage review of the extensions and PDF pipeline test suites - each paired with a new regression test.

- Fixed `prodockit pdf`'s CLI command showing a raw, unhandled traceback instead of a clean `Error: ...` message when `pdf_source_bundle` was enabled and the underlying `git`/`weasyprint` invocation failed - `SourceBundleError` wasn't in the CLI's caught exception tuple.
- Fixed `prodockit.headings` numbering a skipped heading level (e.g. h1 followed directly by h3, or a document starting below h1) with a literal "0" segment (e.g. "1.0.1") - a shallower level with no heading of its own yet is now treated as an implicit first one instead.
- Fixed `prodockit.pdf.mermaid` letting an uncaught `OSError`/`PermissionError` (e.g. a non-executable `mmdc` binary) escape instead of failing just that one diagram gracefully.
- Fixed `prodockit.pdf.source_bundle` crashing the whole bundle build on a file that's valid UTF-8 in the first 8 KiB sniffed to decide "is this text?" but not further in - now skipped like any other binary file instead.
- Fixed `prodockit.pdf`'s generated Lua filter producing broken syntax if a configured math/tex2svg path contained a quote or backslash - both are now escaped.
- Fixed `prodockit.pdf.build_pdf()` having no timeout on the underlying `pandoc`/WeasyPrint invocation, so a hang (e.g. a pathological CSS layout) could block the whole build indefinitely - added a `pandoc_timeout` parameter (default 30 minutes).
- Fixed a back-of-book index term nested more than three levels deep rendering with no extra indent at all, since the generated CSS only defines an

indent step up to level 3 - now clamped to the deepest available indent instead.
- Substantially expanded test coverage across the extensions and PDF pipeline test suites (shared registries, cross-page linking, malformed input, table/index edge cases, icon/rotation/CSS edge cases) - no other functional changes.

1.20 0.6.2 (2026-07-20)

- Docs: fixed a real bug found by checking the live site after 0.6.1 - four spots in `docs/extensions/index-terms.md` / `docs/pdf.md` (plus two more in this same changelog) tried to show the code-styled `\index{}` syntax as literal example text using *inline* backticks around a hierarchical, code-styled path. Confirmed directly this doesn't work the way it does for the plain syntax - the code-styled pattern has to run before Python-Markdown's own backtick handling (see 0.6.0's own entry below), so inline backticks don't protect it, and the live site was rendering a raw internal Python-Markdown placeholder string instead of the intended literal text. Moved each one to a fenced code block (already documented as the safe way to show this syntax) or reworded to avoid the literal example entirely.
- No functional changes.

1.21 0.6.1 (2026-07-20)

- Docs: `prodockit.index` (new in 0.6.0) was missing from `README.md` - and so from PyPI's own project page - entirely: added it to the "Status" line and the extensions table, and mentioned `pdf_include_index` alongside `prodockit.pdf`'s other PDF-only features. Also added it to `pyproject.toml`'s own `description` (PyPI's summary line) and `src/prodockit/__init__.py`'s module docstring, both of which had the same gap.
- No functional changes.

1.22 0.6.0 (2026-07-20)

- New `prodockit.index` extension: mark a term inline with `\index{Term}` for a traditional, PDF-only back-of-book index (browser/Ctrl-F search covers this on the live website, so there's no equivalent there) - the term displays inline exactly as written and is marked for indexing in one go, no separate "definition" step. Needed its own extension rather than the usual `attr_list` marker convention every other prodockit extension uses - confirmed directly plain `attr_list` can't wrap arbitrary inline text in a span on its own.
    - **Sub-entries**: `\index{Parent!Child!Grandchild}` (up to three levels deep in practice, matching LaTeX `makeidx`'s own `\index{primary!secondary!tertiary}` convention) nests related entries together instead of listing every term flat.

    - **Code-styled terms**: backticks around the last segment - or, combined with sub-entries, around just the last segment of a hierarchical path - mark a command/code term: it displays inline in a real `<code>` element, and the generated index entry renders the same way.
    - A term can be a markdown link or contain nested emphasis/code - confirmed directly neither needs special handling, since a term isn't exempted from Python-Markdown's own later inline-pattern passes the way `\ref{id}` / `\cite{id}` / `\gls{id}` are.
- New `prodockit.pdf.index`: the two-pass build (a term's own page number can only be known once WeasyPrint has already laid the PDF out once) behind `pdf_include_index` / `pdf_index_title` (both off/unset by default) - a traditional, two-column, letter-headed index page (matching this project's own cover page hero graphic colour), alphabetised ignoring leading punctuation (so `--set-upstream option` files under "S", not a separate symbols section), with consecutive pages collapsed into an en-dash range (`67–70`). Requires the new optional `pymupdf` dependency - `pip install prodockit[index]`.
- Fixed a real bug found while writing tests: code-styling a non-last segment of a hierarchical term - never a supported combination, but this shouldn't have corrupted anything either - used to leak a raw Python-Markdown internal stash placeholder into the generated index instead of failing gracefully - a real rendered PDF would have shown a nonsense category label instead of "Git".

1.23 0.5.0 (2026-07-19)

- New `prodockit.pdf.source_bundle`: bundles every text/source file `.gitignore` doesn't exclude into a separate `source_bundle.pdf` at a project's own top-level directory - 8pt Courier, wrapped lines, each file starting its own page, a running header (`site_name` on the left, that page's own file path on the right), and a "Page N of M" footer. Off by default; set `pdf_source_bundle = true` under `[project.extra]` to turn it on. Independent of the rest of `prodockit.pdf` - there's no Markdown involved, so it skips Pandoc entirely and hands a small, self-contained HTML document straight to WeasyPrint. File discovery shells out to `git ls-files --cached --others --exclude-standard` rather than reimplementing `.gitignore`'s own matching rules; text/ binary filtering is content-based, not by file extension.
- Docs: this site's own header now shows a PDF download icon next to "view" (an `overrides/partials/actions.html` override, linking to that page's own per-page PDF) instead of a "Download this page as PDF" text link at the top of the page - removed from every page that had one. Since the new icon is template markup rather than Markdown content, it also no longer shows up inside the PDF itself (no `.web-only` CSS trick needed, unlike the link it replaces).

1.24 0.4.2 (2026-07-19)

- Docs: matched more of this site's own theme config to prodockit-userguide's - the header's repo link now shows the actual GitHub logo instead of Zensical's default Git icon; the "View source of this page" button now shows an eye icon instead of a generic file icon; every admonition (e.g. the "tip" callout in `citations.md`) now uses the same custom FontAwesome icon set userguide uses instead of Zensical's own bundled defaults - this also feeds into `prodockit.pdf`'s own admonition icons, so PDF output picks it up too; added the matching theme features userguide already had (`content.tabs.link` in particular actually affects this project's own tabbed content); and swapped the palette toggle icons to match userguide's own light/dark convention.
- No functional (Python package) changes.

1.25 0.4.1 (2026-07-18)

- Docs: reworked this site's own chrome to match prodockit-userguide's - a new split hero cover page ("Home"), reusing that project's own logo/ favicon/ illustration assets; top-level nav moved to a top tab bar with the right-hand page TOC merged into the left sidebar instead; the previous cover page's own prose moved to a new "Introduction" page.
- Fixed a real bug found along the way: `zensical.toml`'s own `copyright` was a triple-quoted, multi-line TOML string - `prodockit.pdf` substitutes it verbatim into a CSS `content: "..."` string for the PDF's running footer, and the embedded newline silently broke that declaration, dropping the whole footer with no error (this site's own PDF footer had no copyright text at all). Fixed to a single-line string, and switched `&copy;` to a literal `©` character - a CSS content string doesn't decode HTML entities either.
- Fixed the deploy workflow missing a per-page PDF build step for the new Introduction page (its own "Download this page as PDF" link 404'd), and that page's leftover PDF link (still the old cover page's, pointing at the whole-site PDF) to the same per-page convention every other content page already uses.
- No functional (Python package) changes.

1.26 0.4.0 (2026-07-18)

- New `pdf_double_sided` option: a duplex-printing layout for book/ handbook-style documents printed and bound on both sides. Verso (left-hand) and recto (right-hand) pages mirror their header/footer content and page margins (new `pdf_margin_inner` / `pdf_margin_outer`, replacing `pdf_margin_left` / `_right` in this mode) via CSS Paged Media's `@page :left` / `:right` selectors - chapter title and page number always on the outer, fore-edge corner; site name and copyright always on the inner, spine-side corner, whichever physical side that is for a given page. Every numbered heading now starts its own recto page (`break-before: recto`,

auto-inserting a blank page as needed - confirmed directly this needs no Python-side page-counting logic at all), and a `prodockit-table-rotated` landscape page's own rotation direction now alternates by its final page position (270 degrees on recto, 90 on verso - the spine sits on the opposite physical side either way).
- New `recto_title` front matter key: overrides a page's own running header text with a shorter title, from the *next* page onward (the heading's own page still shows its full title - confirmed directly this is a consequence of CSS `string()`'s "first value on this page wins" default policy) - useful for a chapter title too long to fit comfortably in the header, with or without `pdf_double_sided`.
- Off by default: a single-sided build is completely unchanged.

1.27 0.3.1 (2026-07-18)

- Docs: renamed `glossary.md`'s heading to "Acronyms and Glossary" and `citations.md`'s to "Citations or References" (and their matching nav labels); added a flow diagram to `bibliography.md`'s Requirements section (and fixed a real, unrelated gap found along the way - this docs site had no Mermaid `custom_fences` config at all, so a plain ```` ```mermaid ```` fence never rendered as a diagram anywhere on the site); switched the citation-style example to `harvard-cite-them-right.csl`; added an admonition pointing from `prodockit.citations` to `prodockit.bibliography`; and noted `prodockit.bibliography`'s own independent Pandoc invocation in `prodockit.pdf`'s "Limitations and workarounds".
- Docs: updated `README.md` (and so PyPI's own project page description) to include `prodockit.tables` / `prodockit.bibliography`, and to mention sideways tables/ `.web-only` / `.pdf-only` under PDF generation - it had gone stale since both extensions shipped in 0.3.0.
- No functional changes.

1.28 0.3.0 (2026-07-18)

- New `prodockit.bibliography` extension: an alternative to `prodockit.citations` for a `.bib`-backed reference list instead of a hand-authored one. Define sources in a BibTeX/BibLaTeX `.bib` file, cite them with the same `\cite{id}` syntax, and get the resolved citation text and a full, auto-generated reference list formatted in any Citation Style Language (CSL) style (APA, IEEE, Harvard, ...) via Pandoc's own `--citeproc` - confirmed directly against real Pandoc output rather than reimplementing citation formatting, and rejected an actual LaTeX/biblatex toolchain as a new hard dependency along the way. Makes `pandoc` a required dependency for this extension specifically, including for a website-only build with no PDF. New `docs/extensions/bibliography.md` includes a "References and Bibliography" comparison of this,

`prodockit.citations`, and what `prodockit-template`/`prodockit-userguide` currently do.
- New sideways (90-degree anticlockwise) tables in the PDF: wrap a table and its own caption in `<div class="prodockit-table-rotated" markdown="1">` to print it on its own landscape-sized page(s), spanning multiple pages with a repeated heading row exactly like any other table. Confirmed directly that a CSS `transform: rotate()` doesn't work for this (clips the table to one page and loses its heading row) - the actual rotation is applied afterwards via a `/Rotate` post-process on the finished PDF (new `prodockit.pdf.rotate` module, new `pypdf` dependency).
- `.web-only` content is now hidden in every PDF build automatically, via `prodockit.pdf.css`'s own always-included stylesheet - no project-side CSS needed any more. `.pdf-only` is documented as a one-line, centrally-sourced snippet instead (`prodockit` has no way to reach into a project's own website stylesheet), in a new "Web-only / PDF-only content" section in the PDF generation docs.

1.29 0.2.0 (2026-07-18)

- New `prodockit.tables` extension: gives a table column a percentage or fixed width via a `width` attribute already attachable to a header cell with `attr_list` - no new syntax. Column-width distribution beyond what's explicitly given is left to CSS's own `table-layout: fixed` algorithm rather than computed in Python. Ships with the matching CSS in `prodockit.pdf`'s generated stylesheet, and documents the equivalent rule a project's own website theme needs (see the new Tables docs page).
- New `prodockit pdf --markdown-file`/`-m` option: builds a PDF from a single markdown file instead of the whole `nav`, using the same `zensical.toml` settings as a full build.
- `prodockit.pdf`'s generated table CSS now draws a full grey 0.5pt grid - outer border and internal row *and* column lines (there was previously no line between columns at all) - and reads a project's own `extra_css` (from `zensical.toml`), so a project-specific `@media print` rule (e.g. hiding a website-only "Download PDF" link/button) also applies in the PDF.
- `prodockit.citations`: a resolved `\cite{id}` link now always gets `class="prodockit-cite-resolved"` (previously no class at all), matching `prodockit.refs`/`prodockit.glossary`'s existing convention of a stable class for both the resolved and unresolved case.
- Docs: added a "CSS hooks" section to `refs.md`/`citations.md`/`glossary.md` (`headings.md` already had one), documenting every class/attribute each extension itself emits; replaced the docs site's "edit this page" link with "view this page" (a `content.action.view` link to the raw source rather than a GitHub edit form); added a whole-site PDF download button on the front page and a per-page download link on every other page, both built via the new `--markdown-file` option above.

- Fixed `prodockit.__version__` reporting a stale `"0.10.0"` (left over from before the `zendoc` → `prodockit` rename) instead of matching this package's actual, `pyproject.toml`-declared version.

1.30 0.1.1 (2026-07-17)

- Docs: reworded the package intro on the docs site and README (dropped the `pymdown-extensions` comparison, added a mention of the website macros and a one-line "kit for professional documentation" summary) - no functional changes.

1.31 0.10.0 (2026-07-15)

- New `prodockit.zensical_macros`: Jinja variables/macros for Zensical's own macros plugin - `{{ word_count }}` (site-wide prose word count, excluding the cover page and any page flagged `exclude_from_word_count: true`), `{{ repo_url }}` (git-detected repository URL), `{{ site_name }}`, and `heading_counter_reset(page)` / `reference_style()` / `acronym_style()` / `glossary_style()` macros. Add it alongside a project's own `macros.py` via `zensical.toml`'s `modules = ["prodockit.zensical_macros"]` - or use it alone if the project has no macros of its own.
- New `prodockit.wordcount`: the generic prose word-count utility (`count_words()` / `compute_word_count()`) behind both `prodockit.pdf`'s `{WORDCOUNT}`-style cover-page use and `prodockit.zensical_macros`' `{{ word_count }}` - previously duplicated independently by each downstream project needing both.
- New `prodockit.settings`: `flatten_nav()`, `heading_numbering_enabled()`, and `reference_style_values()` - the `project.extra.*` reading shared by `prodockit.pdf.config` and `prodockit.zensical_macros`, so the two agree on one set of fallback defaults instead of each hand-maintaining its own copy. `prodockit.pdf.config.build_pdf_from_zensical_config()` now uses these too (previously inlined), and its `pdf_math_dir` setting is now created automatically if configured to a directory that doesn't already exist (matching the auto-detected default's existing behaviour).

1.32 0.9.0 (2026-07-15)

- New `prodockit pdf` command: builds a complete PDF with no Python required, reading everything - nav, docs directory, fonts, page size, and all PDF-specific settings - from the project's own `zensical.toml`, the same way `zensical build` / `zensical serve` do. Installing `prodockit` now registers a `prodockit` console script (`pip install prodockit` is enough - no separate build script to write). See the new

`prodockit.pdf.config` module (`build_pdf_from_zensical_config()`) for the config-to-`build_pdf()` orchestration this wraps: nav-tree flattening, per-page `is_appendix` front-matter detection, and auto-detection of a local `mmdc` (Mermaid) binary and MathJax `tex2svg` script, so a typical project needs no extra configuration beyond what it likely already has.
- `build_pdf()` gained `include_table_of_contents` / `table_of_contents_title` parameters (both used automatically by `prodockit pdf`): a generated table of contents is now inserted by default, right after a cover page if one is marked `is_index=True`, or at the very start otherwise.
- Rewrote the PDF generation docs page around the `prodockit pdf` command as the primary, and for most projects only necessary, way to use `prodockit.pdf` - `build_pdf()` and the individual pipeline pieces are now documented as the advanced, scripting-your-own-pipeline path.

1.33 0.8.0 (2026-07-15)

- New `prodockit.pdf.build_pdf()`: a one-call convenience wrapper around the rest of `prodockit.pdf` - hand it a list of already-rendered pages (`prodockit.pdf.Page`) and where to write the PDF, and it fixes up each page's HTML, generates the Lua filter and CSS, concatenates everything, and runs `pandoc`/WeasyPrint for you. Takes `output_path` (the PDF's own destination path) plus font/page-size/margin/header-footer/reference- style/numbering/math parameters, all with sensible defaults. Raises the new `prodockit.pdf.PdfBuildError` (with the underlying `pandoc` exit code and stderr attached) if the build fails, rather than failing silently. `prodockit.pdf.html`/`.lua`/`.css`/`.icons`/`.mermaid` remain directly importable if you need more control over how the pieces fit together.
- Rewrote the PDF generation docs page around `build_pdf()` as the primary documented way to use `prodockit.pdf`, leading with a short, practical quick-start example rather than the implementation-level detail of how Pandoc/WeasyPrint's own quirks are worked around (that detail is still there, now further down, for anyone who wants it).

1.34 0.7.0 (2026-07-15)

- New `prodockit.pdf`: a Pandoc/WeasyPrint pipeline for building a standalone PDF from Zensical-rendered HTML - not a Python-Markdown extension (no `markdown.extensions` entry point), a plain function library, since a PDF build pipeline isn't a Markdown syntax extension:
  - `prodockit.pdf.html`: `fix_up_page_html()` and link/anchor/image helpers
  - fixes up one page's already-rendered HTML for Pandoc's own reader/writer quirks (attribute loss on `<p>`, raw `<svg>` not surviving the round

trip to WeasyPrint, footnote/caption structural mismatches, cross-page link rewriting for a concatenated multi-page PDF, and more).
    - `prodockit.pdf.lua`: `build_lua_filter()` - chapter/appendix numbering, caption chapter-prefix numbering, tabbed-set reconstruction, and MathJax pre-rendering, generated as a parameterized Lua filter.
    - `prodockit.pdf.css`: `build_css()` - the compiled CSS a PDF needs on top of a project's own website stylesheet, including WeasyPrint-specific page-break tuning for headings, paragraphs, tables, code blocks, figures/captions, admonitions, and grid cards.
    - `prodockit.pdf.icons` / `prodockit.pdf.mermaid`: admonition icon resolution and Mermaid diagram pre-rendering, as standalone helpers.
- Fixed a real bug found while writing tests: the iframe→"Watch Video" admonition link builder stripped the video id from every single conversion (a replace-then-split ordering removed the just-added `?v=...` too) - now produces a working YouTube watch link.
- No formal, versioned public API surface yet (see prodockit-extension#7) - import whatever's needed directly, the same informal way as the rest of this package.
- New dependency: `beautifulsoup4` (>= 4.12).
- Broadened the package's own description: prodockit is now framed as a family of extensions for Zensical needed for professional and academic documentation, rather than "Python-Markdown extensions" specifically - `prodockit.pdf` isn't one, and the framing was due to broaden anyway now that PDF generation is in scope alongside cross-references/citations/glossary.

1.35 0.6.0 (2026-07-14)

- `prodockit.headings`: new `numbering="continuous"` option (Zensical only) - `h1` numbering carries on from wherever the previous nav page left off, instead of restarting at 1 on every page. Fixes `\ref{id}` showing the wrong number for a heading on a different page (it previously always showed that page's own per-document number, not the number actually displayed on the page - see zendoc-template#89).
- New `appendix_attr` option (default `is_appendix`): a page whose front matter sets this flag is numbered with a letter instead - "A", "A.1", "A.1.1" - and doesn't consume a number from the numeric sequence, so later pages aren't left with a gap. Letters are assigned sequentially in nav order.
- New public `prodockit.headings.prescan(appendix_attr="is_appendix")` function: returns the same `(start_counts, appendix_letters)` pre-scan `HeadingsExtension` uses internally, for a consuming project's own build tooling (e.g. a template macro driving a presentational CSS counter-reset) to stay in sync automatically rather than re-deriving the same page-order/heading-count logic independently.

1.36 0.5.1 (2026-07-14)

- `prodockit.glossary`: a resolved `\gls{id}` now always renders with `class="prodockit-gls"` (previously it had no class at all), matching `prodockit.refs`' always-present base class. The unresolved case now renders `class="prodockit-gls prodockit-gls-unresolved"` (previously just `prodockit-gls-unresolved`, missing the base class), so a stylesheet has one stable hook (`.prodockit-gls`) regardless of resolution state, with `.prodockit-gls-unresolved` layered on top only when needed.

1.37 0.5.0 (2026-07-14)

- New `prodockit.glossary` extension: define a term once via `attr_list` (an id plus a `data-term` short display string), then insert it by id from anywhere with `\gls{id}`, which resolves to the term's own text, linked to its definition - e.g. `\gls{css}` → `CSS`. Unlike `prodockit.citations`' `\cite{id}` (which generates new bracketed citation text), `\gls{id}` inserts the term's own registered text in place - closer to LaTeX's `glossaries` package.
- One shared `GlossaryRegistry` covers both acronym-style and glossary-style entries - they're the same kind of thing (an id with a short display text), so acronym and glossary pages can reference each other, or be referenced from any other page, with no special wiring.
- Supports forward references within a document, an `unresolved` marker (`?` by default) for an unknown id, and the same automatic Zensical cross-page registry sharing and nav pre-scan (for citing/using a term before its defining page has been converted) that `prodockit.citations` got in 0.4.0.
- Refactored the nav pre-scan logic (previously private to `prodockit.citations`) into a shared, generic `prodockit._zensical.preseed_attr_from_nav` helper, since `prodockit.glossary` needed the identical scan.

1.38 0.4.0 (2026-07-14)

Fixes found migrating a real multi-page site's references page to `prodockit.citations` for real - all discovered by actually building a real multi-page site, not just single-document tests:

- **Fixed a real correctness bug**: `prodockit.refs`/`prodockit.citations` were emitting a bare `#id` fragment for *every* resolved link, including a cross-page one - which only works by coincidence in a single concatenated PDF document, but 404s on an actual multi-page website (an `#id` fragment only navigates within the *current* page). Both now emit a real relative link (e.g. `references.md#id`, correctly adjusted for the citing page's own directory depth) when the target is on a different page, which Zensical already knows

how to rewrite into the right clean URL - the same way a hand-typed cross-page Markdown link already works.
- New: `prodockit.citations` pre-scans every page in a Zensical build's nav for citation definitions before any page is actually converted, so citing a source *before* it's defined - the common case, since a references page is usually kept at the end of nav as an appendix - resolves correctly in a single `zensical build` pass, rather than only working from `zensical serve`'s live-reload. New `CitationRegistry.preseed()` method backs this; a real registration always supersedes a preseeded stub.
- `RefsExtension` gained a `source` option (mirroring `HeadingsExtension`'s), needed for the same-page-vs-cross-page link decision above.
- Fixed the nav pre-scan matching a citation-definition attr_list example shown literally inside a fenced code block in documentation - it now skips fenced content, the same protection `CitationDefTreeprocessor` already gets for free from the real Python-Markdown parser.

1.39 0.3.0 (2026-07-14)

- New `prodockit.citations` extension: define a source once via `attr_list` (an id plus a `data-cite-text` short display string), then cite it by key from anywhere with `\cite{id}` (or `\cite{id1,id2,...}` for multiple), auto-generating a bracketed, linked citation - `[Skoulikari, 2023]` - instead of hand-typing the link and text at every citation site.
- Supports forward references within a document, an `unresolved` marker (`?` by default) for an unknown key, and the same automatic Zensical cross-page registry sharing (with soft-fail on key collisions) that `prodockit.headings` / `prodockit.refs` got in 0.2.0.
- Auto-generating the references page's own listing from structured bibliographic data isn't built yet - see the extension's docs for the current scope.
- Fixed the `zensical.toml` installation examples in the docs: nested `[project.markdown_extensions.prodockit.headings]` tables don't work (Zensical only hoists the `pymdownx` / `zensical` namespaces that way) - the quoted-key form (`[project.markdown_extensions."prodockit.headings"]`) is required.

1.40 0.2.0 (2026-07-14)

- `prodockit.headings` / `prodockit.refs` now share their registry automatically under Zensical, without any explicit `registry` / `source` configuration: each extension detects Zensical's per-page rendering context and derives a stable `source` from the page's own path, fixing cross-page `\ref{id}` references not resolving.
- A heading id collision across two different sources, when detected via this automatic Zensical sharing, now logs a warning and keeps the first

registration instead of raising `DuplicateIdError` - so two unrelated pages that happen to share a heading title (e.g. both have an "Overview" section) no longer break the build. Explicitly-shared registries (the manual multi-page pattern) still raise on a collision, unchanged.
- Fixed an extension-ordering bug: `prodockit.headings` and `prodockit.refs` now find and share each other's registry regardless of which order they're listed in - previously, only `prodockit.headings`-then-`prodockit.refs` worked reliably, and Zensical's own TOML-to-extension-list conversion doesn't preserve list order at all.

1.41 0.1.0 (2026-07-14)

Initial release.

- `prodockit.headings`: heading ids and hierarchical section numbering, backed by a shared `IdRegistry`.
- `prodockit.refs`: `\ref{id}` section cross-references, resolving to the target's current section number, including forward references within a document and across a shared registry.
- Documentation site built with Zensical, published at buckwem.github.io/prodockit-extension.