Table of Contents

1. Cross-references

`prodockit.refs` creates links to headings elsewhere in your documentation. Each link shows the heading's current number and name, such as “1.1 Configuration”, so you do not have to update it when sections move.

Use `\ref` for a link that works on the website and in the PDF. Use `\autoref` when the PDF should also show the target's page number.

1.1 Enable the extension

Enable it in `zensical.toml`:

```
[project.markdown_extensions."prodockit.refs"]
```

1.2 Reference a heading

Reference any heading's id with `\ref{id}`:

Markdown

```
# Introduction {: #intro }

See \ref{intro} for background.

## Background
```

Result

Introduction

See 1 Introduction for background.

Background

The link takes the reader to the `Introduction` heading. Its number and name update automatically if the document changes.

1.3 Configure cross-references

1.3.1 Choose the missing-reference text

An unresolved reference displays `??` by default. Set `unresolved` if your project uses a different marker:

```
[project.markdown_extensions."prodockit.refs"]
unresolved = "MISSING"
```

`source` is the only other TOML setting. It identifies the current page, but Zensical detects it automatically; leave it unset in `zensical.toml`.

1.3.2 Reference a heading before it appears

A reference to a heading defined *later* in the same document resolves correctly:

```
See \ref{background} below.

## Background {: #background }
```

1.3.3 Fix a missing reference

If the heading id is missing or mistyped, `\ref{id}` displays `??` instead of a link. Check that the text inside the braces exactly matches the id on the heading.

A heading marked `unnumbered` still works as a link. Because it has no section number, the link displays only the heading name.

```
# Cover Page {: .unnumbered #cover-page }

See \ref{cover-page}.
```

renders `\ref{cover-page}` as `Cover Page`, linked to `#cover-page`.

1.3.4 Include a page number in the PDF

`\ref{id}` and `\autoref{id}` render exactly the same text - the target's number and name. The difference is that `\autoref{id}` also carries the target's **page number** in the PDF, which is what a reader holding a printout needs and what a website reader has no use for:

Markdown

```
Configuration is covered in \autoref{configuration}.
```

Website

Configuration is covered in 1.1 Configuration.

PDF

Configuration is covered in 1.1 Configuration on page 12.

The " on page N" suffix comes from prodockit.pdf's own stylesheet, so it appears only in the PDF - a page number on a scrolling website would be meaningless. Nothing to enable: build the PDF and it is there.

Which to use is a per-reference decision rather than a project-wide setting: use `\autoref{id}` where a printed reader needs to turn to something, and `\ref{id}` where the extra "on page N" would just be noise.

An appendix needs nothing special - its letter is already the first segment of its number, so `\ref{terms}` renders "A.1 Terms".

1.3.5 Reference a figure or table

`\ref{id}` also resolves a captioned figure or table, rendering its label:

```
![Component Model](assets/images/component-model.png)
{ width="100%" }
/// figure-caption
    attrs: {id: fig-component-model}

Component Model
///

The components inside the System Context boundary are shown in
\ref{fig-component-model}.
```

which renders as a link reading **Figure 3.1**.

Figures and tables are counted separately, and both restart per page and carry the page's chapter number - the same numbering the caption itself shows, so a reference and the thing it points at always agree.

> ⚠ **The id goes in an `attrs:` option, not `{: #id }`**
>
> Caption blocks take attributes the Blocks API way - an indented `attrs:` line, then a blank line, then the caption text. The `{: #id }` form used on headings, images and table cells **does not work here**: it produces no figure at all, silently, and the `///` lines appear as literal text.
>
> An id containing a colon (`fig:component-model`) also produces no figure, quoted or not. Use a hyphen.

Unlike a heading, a caption reference is **its label alone** - "Figure 3.1", not "Figure 3.1 Component Model". A caption is referred to mid-sentence, where repeating its own words reads as a stutter; a heading's number alone would say nothing about where the reader is being sent, so that keeps its name.

1.4 Reference

1.4.1 Syntax

Syntax	Result
`\ref{<id>}`	The target's current number and name
`\autoref{<id>}`	The same link, plus “on page N” in the PDF

`<id>` is the target heading's id - either one you set explicitly via `attr_list` (`# Introduction {: #intro }`), or the one `toc` derived automatically from the heading text (see prodockit.headings for the exact precedence).

`\ref{...}` is recognised the same way Python-Markdown's own inline syntax is - meaning it's protected inside inline code spans and fenced code blocks, so it's safe to show as a literal example:

````
Type `\ref{intro}` to reference a section.

```
\ref{intro}
```
````

Neither of the two shown above is resolved; both render the literal text.

1.4.2 Zensical settings

Setting	Default	What it controls
`unresolved`	`"??"`	Text shown when an id cannot be found.
`source`	`""` (detected automatically)	Advanced: identifies the current page when using the extension outside Zensical. Leave it unset in `zensical.toml`.

1.4.3 Cross-page references

Under Zensical, cross-page references work automatically when headings and references are enabled. Prodockit reads heading ids and numbers across the navigation before individual pages finish rendering, so a reference can point to a page built later.

Two pages with the same generated heading id produce a warning. Give the headings distinct explicit ids so every destination is stable.

For integration with another Markdown renderer, see Extension integration.

1.5 Customise with a CSS style sheet

`prodockit.refs` always sets a class on the `\ref{id}` link it renders - resolved or not - so a stylesheet has a stable hook either way:

Syntax	State	Class
`\ref{id}`	Resolved	`prodockit-ref`
`\ref{id}`	Unresolved	`prodockit-ref prodockit-ref-unresolved`
`\autoref{id}`	Resolved	`prodockit-autoref`
`\autoref{id}`	Unresolved	`prodockit-autoref prodockit-autoref-unresolved`

An unresolved reference (see Unresolved references above) still gets a `class` either way; style `prodockit-ref-unresolved` distinctly (e.g. a warning colour) to make a broken cross-reference visually obvious. Unresolved references have no destination, so an unresolved `\autoref` does not print a stray page-number suffix in the PDF.

Index

P