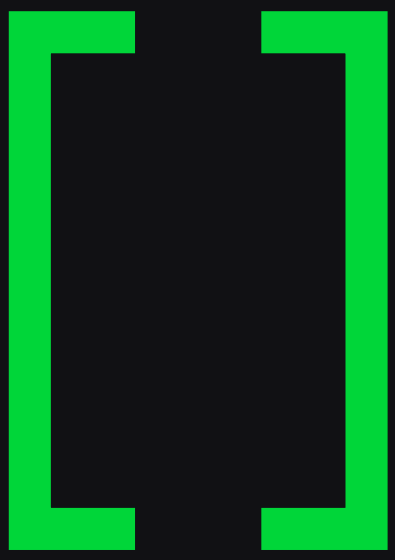
proto[docs]

ARCHITECTURE DOCUMENTATION

Document Every System the Same Way

What a shared template gives your readers & writers, & how to keep it small enough to use.



Matt Hensley · November 2026

proto[docs]

CONTENTS

What's in Here

01 **Every Team Writes It Differently**
Five teams, five layouts

02 **Readers Know Where to Look**
Same sections · Same order

03 **Gaps Show Up on Their Own**
Empty slots · The backlog

04 **Writers Start Faster**
Questions instead of a blank page

05 **Keep the Template Small**
A short core · Sections that apply

06 **Resources**
Further reading

01

Every Team Writes It Differently

Ask five teams to document their services & you get five different documents. Each one makes sense to the person who wrote it.

One team leads with a diagram. Another opens with setup steps & tucks the dependencies under a heading called “Misc.” A third never says who owns the thing. On its own, each page is fine. Side by side, they’re hard to use.

The reader pays for it. Every page is a new layout to learn before they find the answer they came for, & on-call doesn’t have time to learn a layout.

FIGURE 1 **Where each team keeps its dependencies**

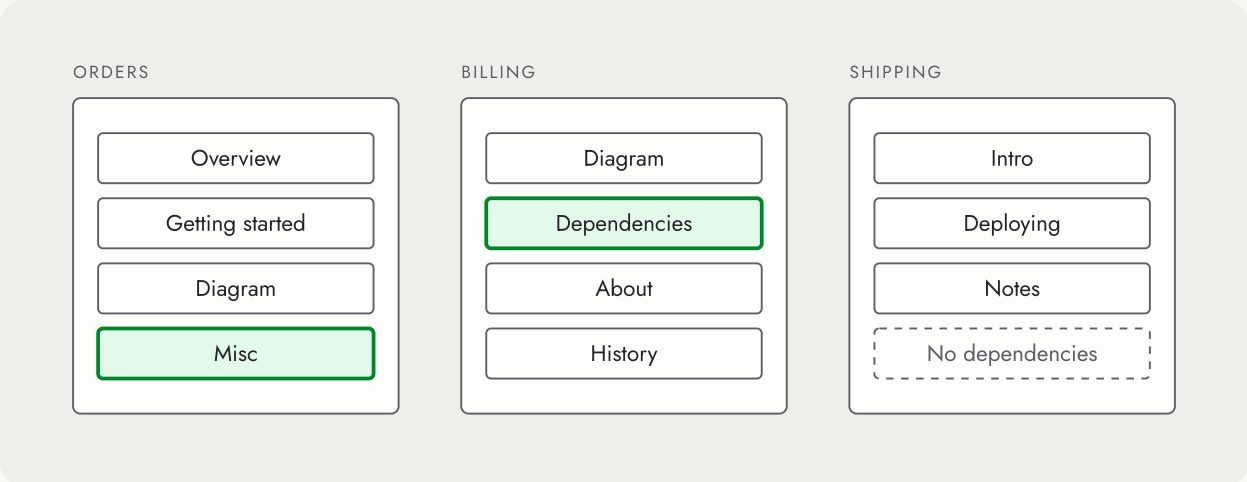


Figure 1. Three teams, three layouts. Green marks where each page keeps its dependencies, when it has them.

02

Readers Know Where to Look

Give every system the same sections in the same order & readers stop searching. They go straight to the part they need.

FIGURE 2 **The same sections in the same order**

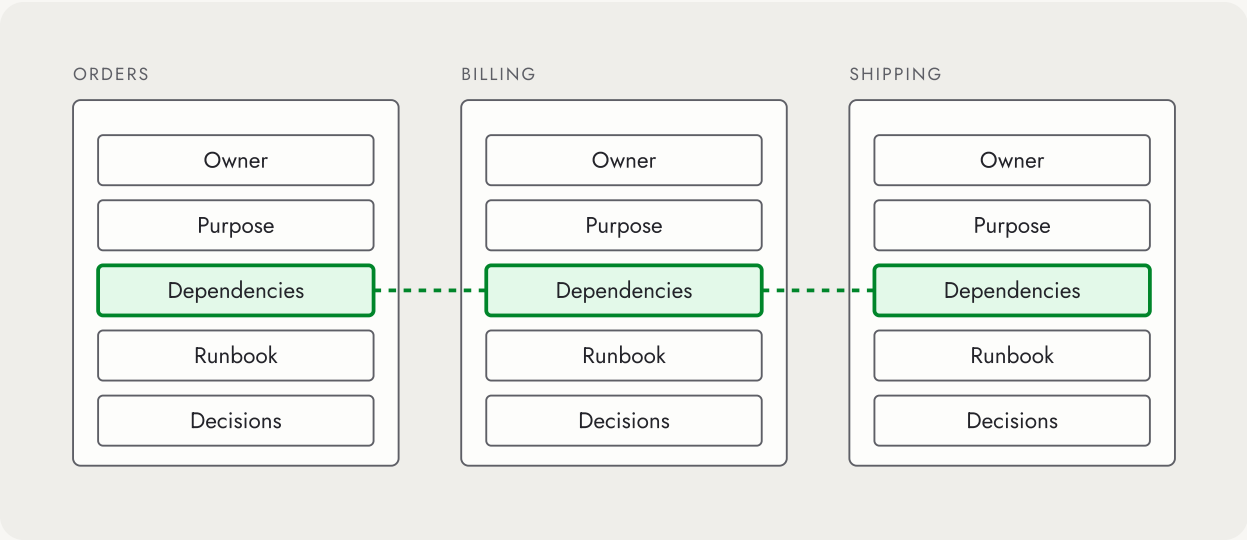


Figure 2. One template for every system. Dependencies are always the third section, even on a page you've never opened.

After the second page, a reader knows the owner is always at the top & dependencies are always third. That holds for a service they've never touched. It holds for people outside engineering too, who read your documentation least often & gain the most from not relearning it.

[ ]

The Architecture Documentation Map covers which documents to write & who reads each one. A template decides what goes inside them, so every system answers the same questions in the same place.

03

Gaps Show Up on Their Own

Freeform documentation hides what's missing. A template turns every missing answer into an empty slot you can see.

FIGURE 3 Systems against template sections

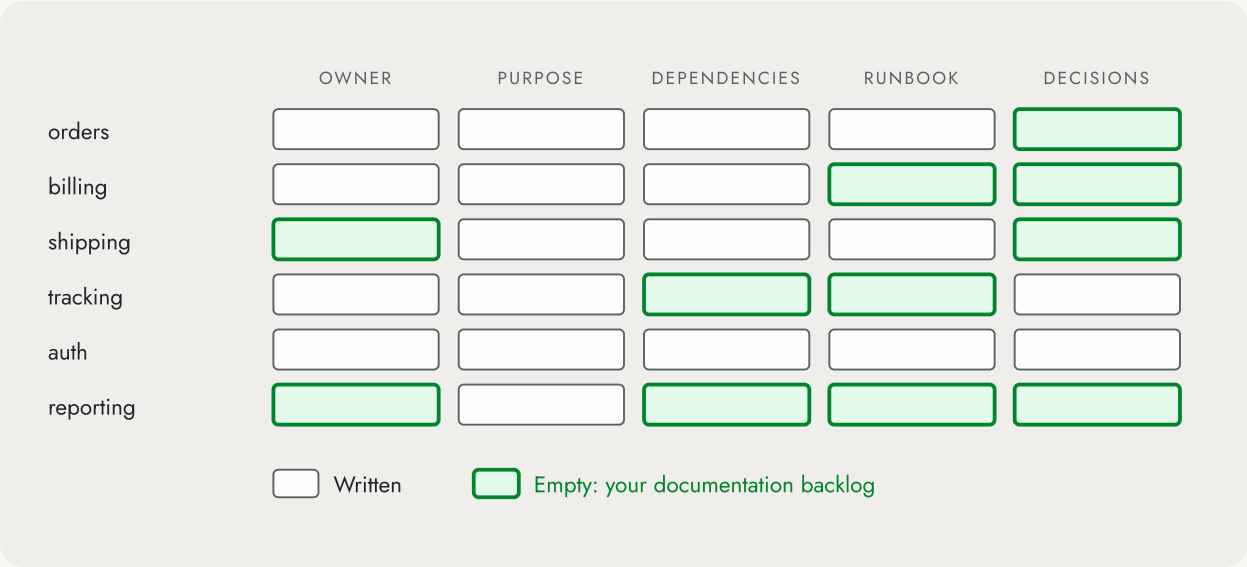


Figure 3. Line up every system against the template & the empty slots are the work left to do.

Without a template, nobody can tell whether billing has no runbook or keeps it somewhere else. With one, the gap is an empty section under a heading everyone recognizes.

The same structure makes documentation checkable. When every page has an Owner section, listing the systems with no owner takes a minute instead of a read through every page.

04

Writers Start Faster

The blank page is the hardest part of writing documentation. A template replaces it with a list of questions.

FIGURE 4 A blank page & a template

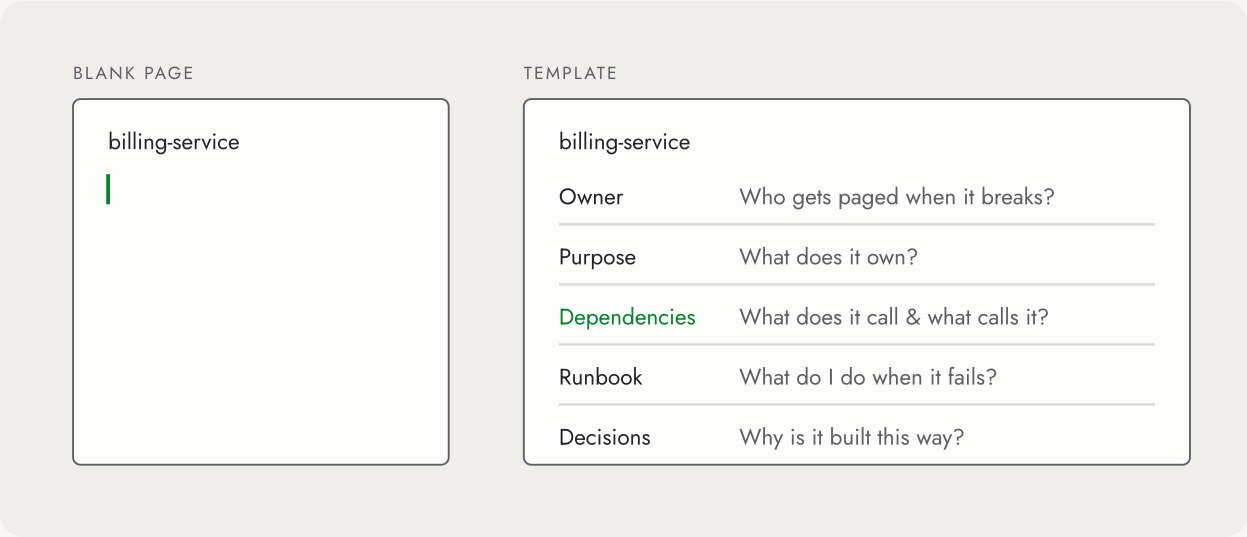


Figure 4. Each section is a question a reader will ask. The writer answers it & moves on.

With a template, the writer doesn't have to decide what to cover. They answer the questions that are already there, in the order they're already in.

Reviews get easier too. A reviewer checks each section against its question, instead of debating what the page should contain. New teams get the house style for free on their first service.

05

Keep the Template Small

A template can fail the same way a freeform page does, by trying to cover everything.

FIGURE 5 **A short core & sections that apply**

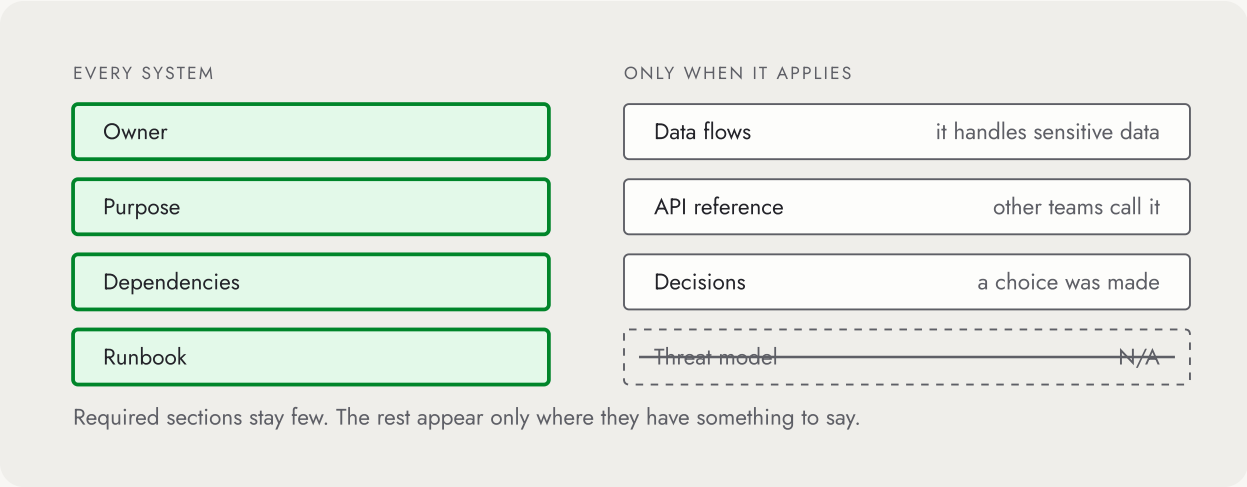


Figure 5. Nobody should have to type N/A to get through a template.

Every section you add is a section someone fills in for every system, for as long as it exists. Make too many required & writers start typing “N/A” to get through it, & readers learn to skip whole pages. Keep a short core every system answers & add sections only where they apply.

Start with the questions people already ask in chat. If nobody asks it, it doesn’t need a section yet.

06

Resources

Further reading on documentation structure & templates.

arc42: a template for architecture documentation

arc42.org

Diátaxis: the four documentation modes, by Daniele Procida

diataxis.fr

The C4 model: context, container & component diagrams, by Simon Brown

c4model.com

Documenting Architecture Decisions: the original ADR post, by Michael Nygard

cognitect.com/blog/2011/11/15/documenting-architecture-decisions

The Architecture Documentation Map: which documents to write & who reads each one

prototypd.dev/blog/architecture-documentation-map

proto[docs]

prototypd.dev/docs