Making Documentation AI-Ready

Preparing Your Docs for the LLM Era

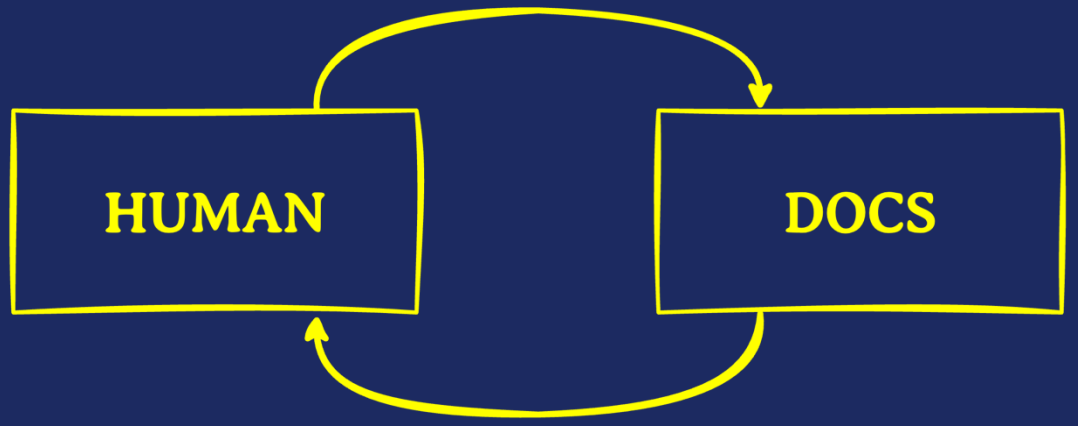
Edvin Teskeredzic

SENIOR AI SOFTWARE ENGINEER @ INFOBIP

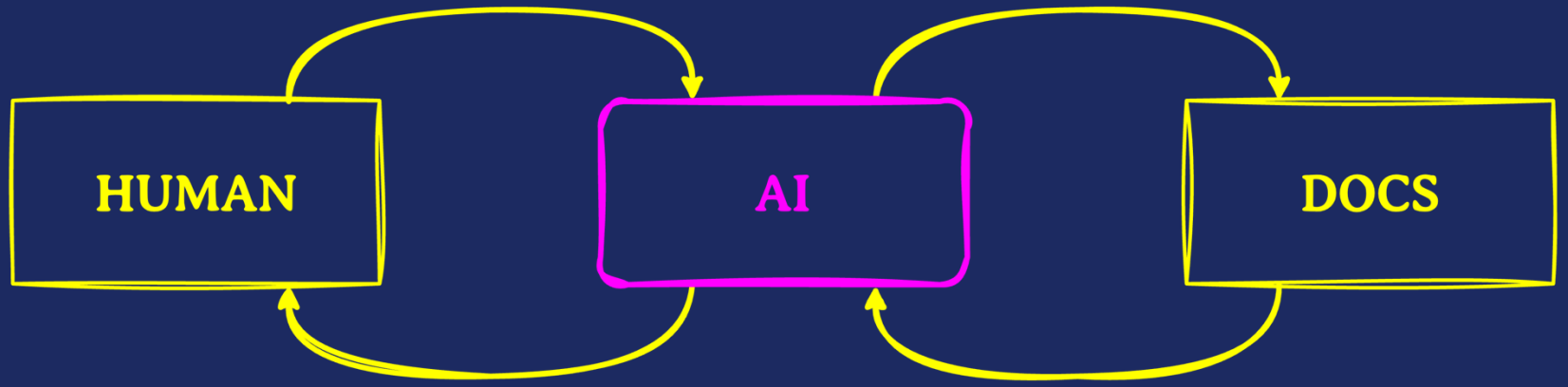
Edvin.Teskeredzic@infobip.com

Good docs used to be **scannable**. Now they also need to be ***extractable***.

The Reader Has Changed



The Reader Has Changed



AI systems retrieve chunks, inspect page structure, follow links...

... they do not read like humans do!

The Adoption-Trust Gap

84%
Use or plan to use AI tools

30%
Trust AI output

66%
Spend more time fixing almost-right AI code

46%
Actively distrust AI output

Developers use AI. They don’t trust it.

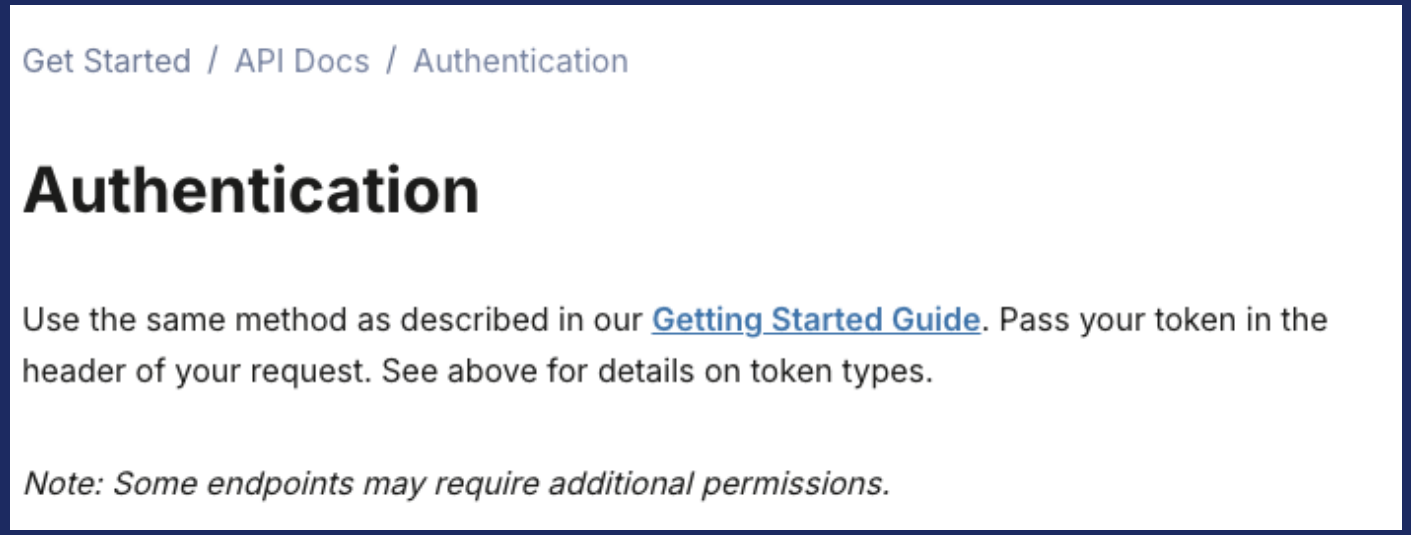
Better docs help close that gap.

How AI Actually Reads Your Docs

1. Retrieval chunks
2. Full-page crawl
3. DOM
4. Machine-readable layers
5. Screenshots

Your docs are consumed through at least five pathways!

The Failure Mode



Get Started / API Docs / Authentication

Authentication

Use the same method as described in our Getting Started Guide. Pass your token in the header of your request. See above for details on token types.

Note: Some endpoints may require additional permissions.

- Which token? What header? Which method?
- ”See above” means nothing in a retrieved chunk
- “Some endpoints” – which ones?

The problem?
Docs which depend on surrounding context!

Contextual Density Mapping

Enough local context for a machine to interpret a section correctly, without bloating the doc for humans.

Five questions every chunk should answer:

1. What is this? → The subject.
2. Who is it for? → The audience.
3. What scope applies? → Version, product, environment, etc.
4. What does it depend on? → Prerequisites, auth, state.
5. What comes next? → The logical next step.

Technique 1: Context Sandwich

Answer

↓

Orientation

↓

Nuance

Get Started / API Docs / Rate Limiting

Rate Limiting

Our API implements rate limiting to ensure fair usage across all customers. Rate limiting is a common practice in API design that helps prevent abuse and ensures service reliability. The approach we took...

[3 paragraphs of text]

The default rate limit is **100** requests per minute per API key.

Technique 1: Context Sandwich

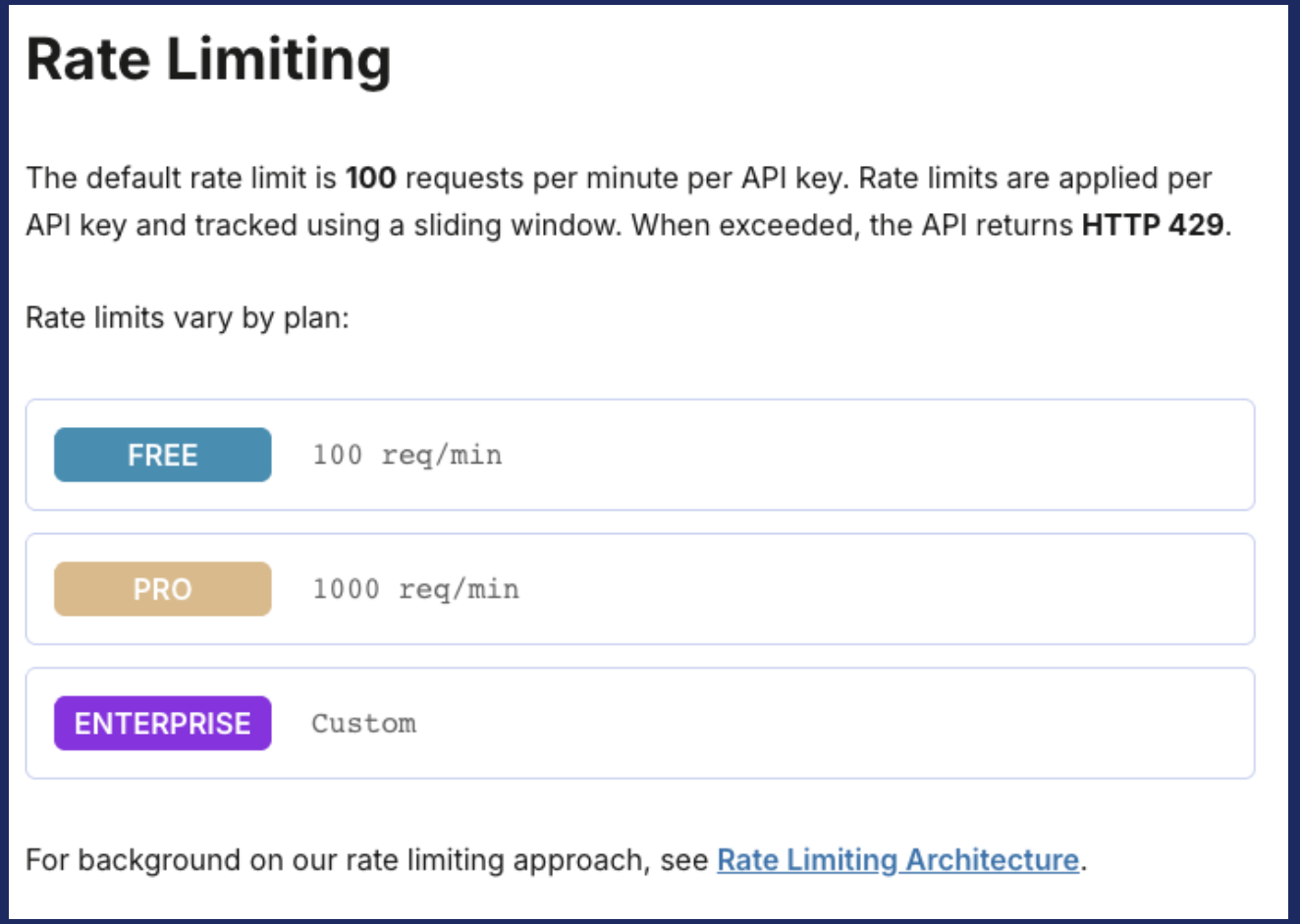
Answer

↓

Orientation

↓

Nuance



Rate Limiting

The default rate limit is **100** requests per minute per API key. Rate limits are applied per API key and tracked using a sliding window. When exceeded, the API returns **HTTP 429**.

Rate limits vary by plan:

FREE	`100 req/min`
PRO	`1000 req/min`
ENTERPRISE	`Custom`

For background on our rate limiting approach, see Rate Limiting Architecture.

Technique 2: Explicit Relationship Mapping

Name the **subject**, the **object**, the **dependency**, the **scope**, and the **next step**.

Never rely on the reader having read the **previous** section.

Before

Sending Messages

Use the Send endpoint as described above. Make sure you've completed all previous steps first. Pass the parameters as shown.

See also: guide.

Technique 2: Explicit Relationship Mapping

Name the **subject**, the **object**, the **dependency**, the **scope**, and the **next step**.

Never rely on the reader having read the **previous** section.

After

Sending Messages

Use **POST /api/v2/messages** to send an SMS message.

Prerequisites:

- Active API key (see authentication)
- A registered sender ID (see sender management)
- The recipient must be a valid E.164 phone number

Required parameters:

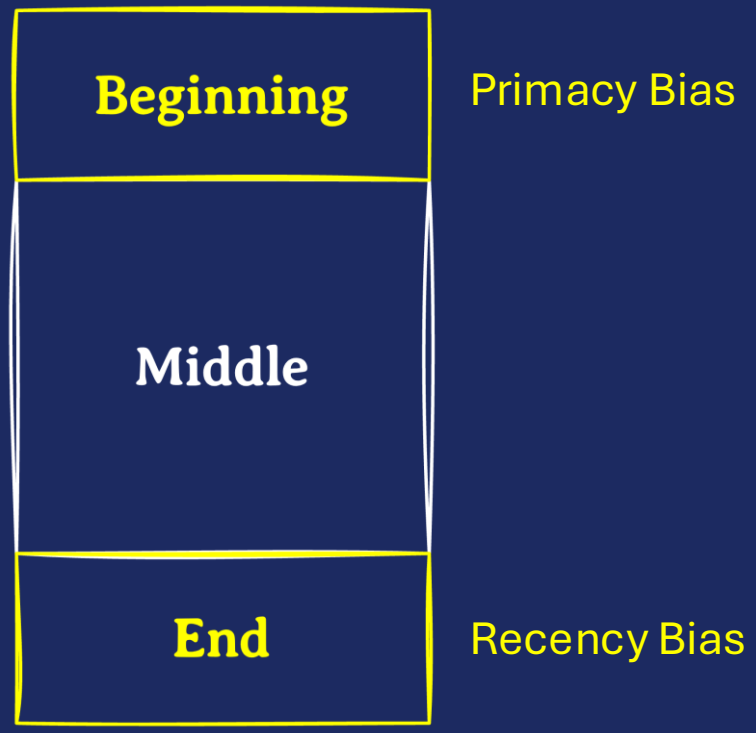
- `to` (string): Recipient phone number in E.164 format
- `from` (string): Your registered sender ID
- `text` (string): Message content, max 1600 characters

Next step: check message delivery status.

Technique 3: Progressive Context Disclosure

- Organize information in layers: essential first, detailed second, edge cases third
- Model performance highest: relevant information at the beginning or end of input – degrades when in the middle
- Context length grows – performance drops

Long context is not a license for bad information architecture



Technique 4: Contextual Anchoring

- Anchor new concepts to previously established ones
- Helps AI models interpret content relationships and map dependencies between concepts

Example

Best practices for phone and email verification guide

Now that you understand what phone and email verification are, what the benefits are for your business, and how to implement each – it's time to look at best practices when implementing these solutions.

1. Anchors new content
2. Signals a local progression
3. Uses clear linkage language

Beyond Prose: How agents see your site

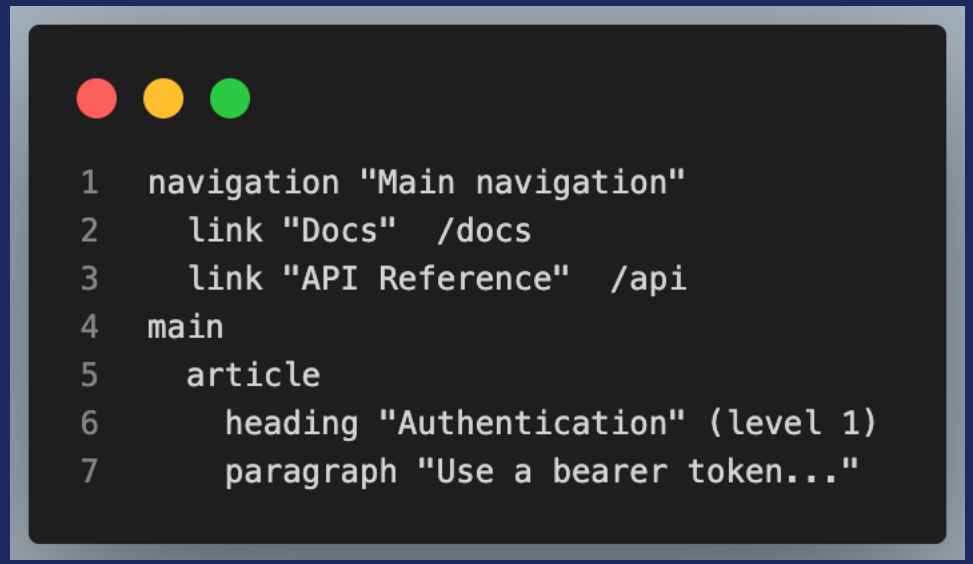
```html
<div class="nav-wrapper">
    <div class="nav-item" onclick="goto('/docs')">Docs</div>
    <div class="nav-item" onclick="goto('/api')">API</div>
</div>
<div class="content-box">
    <div class="title-big">Authentication</div>
    <div class="text">Use a bearer token...</div>
</div>
```

```
generic
    generic "Docs"
    generic "API"
    generic
      generic "Authentication"
      generic "Use a bearer token..."
```

Agent's accessibility tree:
(no roles, no landmarks, no navigable structure)

Beyond Prose: How agents see your site

```html
<nav aria-label="Main navigation">
    <a href="/docs">Docs</a>
    <a href="/api">API Reference</a>
</nav>
<main>
    <article>
        <h1>Authentication</h1>
        <p>Use a bearer token in the <code>Authorization</code> header...</p>
    </article>
</main>
```

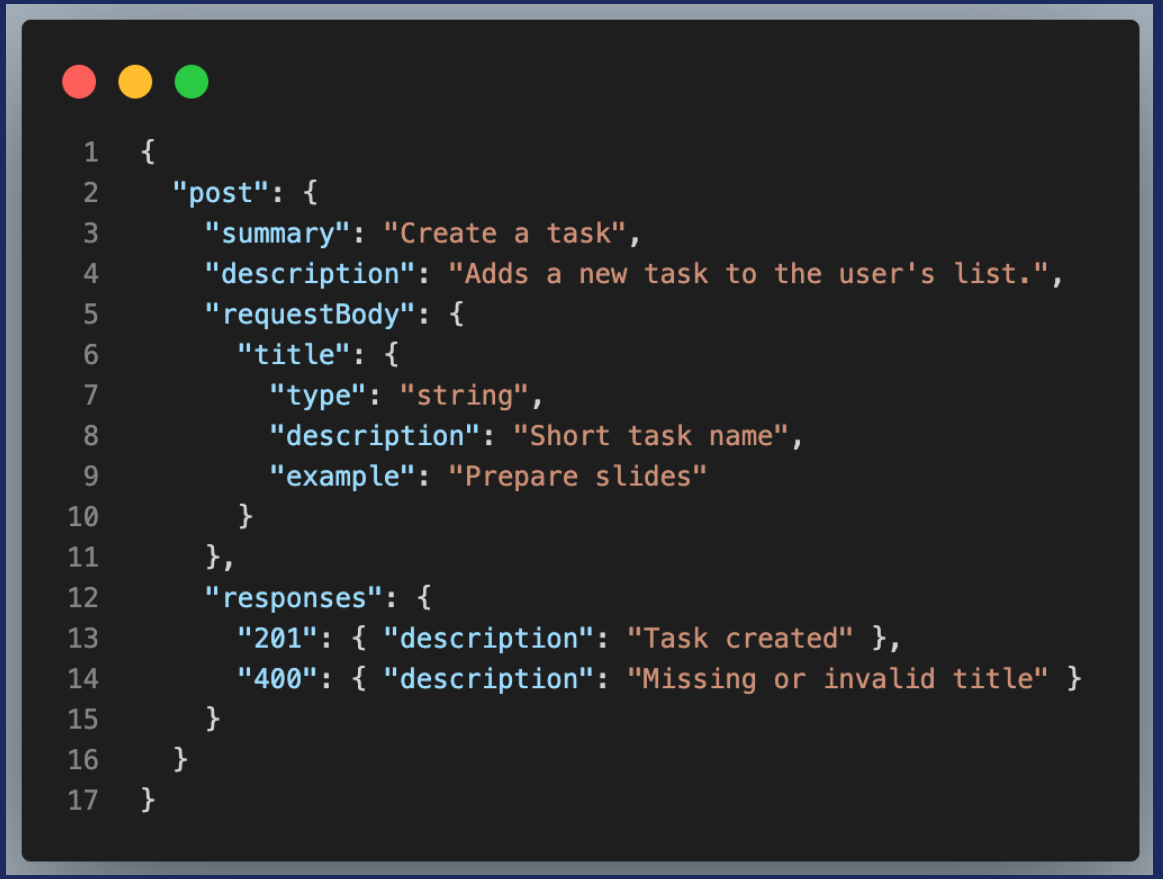


```
navigation "Main navigation"
  link "Docs"  /docs
  link "API Reference"  /api
main
  article
    heading "Authentication" (level 1)
    paragraph "Use a bearer token..."
```

Agent's accessibility tree:
(rich, navigable, meaningful)

Machine-Readable Layers: OpenAPI

- Beyond semantic HTML, provide dedicated machine-readable layers
- **OpenAPI** – de facto standard for the API's formal contract
  - Better OpenAPI descriptions = fewer hallucinated API calls
  - Rely on semantic clues in the **`summary`** and **`description`** fields
  - OpenAPI spec can we converted into an MCP server (see https://github.com/infobip/infobip-openapi-mcp)



```
{
  "post": {
    "summary": "Create a task",
    "description": "Adds a new task to the user's list.",
    "requestBody": {
      "title": {
        "type": "string",
        "description": "Short task name",
        "example": "Prepare slides"
      }
    },
    "responses": {
      "201": { "description": "Task created" },
      "400": { "description": "Missing or invalid title" }
    }
  }
}
```

Machine-Readable Layers: llms.txt

- **`llms.txt`** – a manifest for AI discovery
  - One MD file (**`/llms.txt`**) that gives AI systems a structural map
  - Companion: **`llms-full.txt`** compiles all your docs into a single MD file
  - This is an **emerging** and **optional** convention – if your pages are ambiguous, **`llms.txt`** won't save you

```
# Acme Docs

> Developer documentation for Acme's task management API.

## Core Docs

- [API Overview](https://example.com/docs/api): Authentication, base URLs, and rate limits.
- [Create Task](https://example.com/docs/tasks/create): Request body, responses, and examples.
- [Error Codes](https://example.com/docs/errors): Common errors and how to fix them.

## Optional

- [Changelog](https://example.com/changelog): Recent API updates.
```

Machine-Readable Layers: MCP

- **MCP (Model Context Protocol)** – protocol for AI agents to discover and invoke tools at runtime
- **WebMCP** – turn your websites into an agent interface
  - Two approaches – declarative and imperative
  - Declarative: Add `toolname` and `tooldescription` attrs to existing HTML `<form>` elements
  - Imperative: Register tools via JavaScript for complex or stateful interactions

```
<form toolname="search_docs"
      tooldescription="Search the docs by keyword"
      action="/search" method="GET">
  <input name="query" type="text" required
         toolparamdescription="Search keywords" />
  <button type="submit">Search</button>
</form>
```

Machine-Readable Layers: OpenAPI, llms.txt, and Beyond

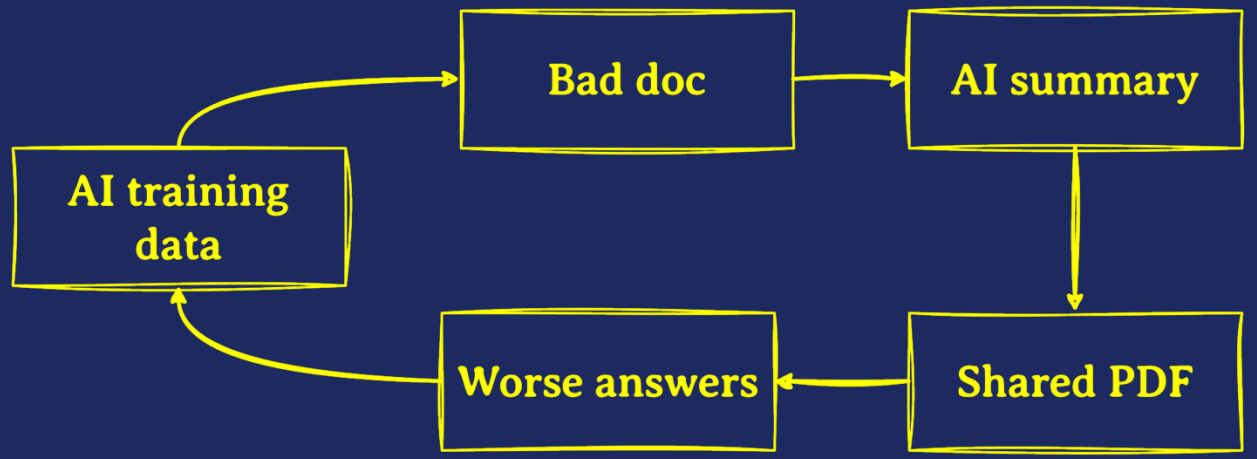
These layers are useful delivery mechanisms.

But they deliver whatever you give them; if the source content is bad, you're just serving bad content faster!

The Governance Problem

- **Shadow docs**: Teams are using AI to generate unofficial guides from your official docs

- These shadow docs:
  - Do not get updated when the source does
  - May contain hallucinations
  - May mix content from different API versions
  - Become the “real” docs for some teams
  - Get fed back into AI tools, amplifying errors



The Audit Checklist

A simple scorecard for any documentation page:

Check	Question
Self-contained	Can a single section answer a question without context from other sections?
Answer-first	Is the key information in the first 2-3 sentences?
Explicitly linked	Do all references name their target?
Scope-labeled	Does the section state product, version, plan, or environment it applies to?
Dependency-declared	Are prerequisites listed, not assumed?
Semantically structured	Are headings descriptive? Is markup semantic?
Governance-tracked	Is there a canonical source? Are shadow docs known and managed?

Wrapping Up

AI-ready docs are docs that remain accurate when read out of order, out of context, or through a machine

- Everything covered today makes docs better for ***everyone***, not just agents
- The unit of change is documentation; we cannot control ***how*** AI reads our docs, but we can control ***what*** it finds

- **Start on Monday:**
    1. **Pick your 10 most visited pages.**
    2. **Run the five-question audit.**
    3. **Ship the improvements!**

Q & A

Q&A

- **Isn’t this just SEO but rebranded?**
  - Not quite – there is overlap, but here we consider chunk survivability, retrieval behavior, and agent interfaces such as (Web) MCP. SEO still matters, while modern agents rely on DOM and accessibility representations.
- **Won’t larger context windows make this irrelevant?**
  - No – long context does not erase problems such as relevance or attention. Context remains a finite resource with diminishing returns (per Anthropic and Academia).
- **If we have OpenAPI, why do we need prose docs?**
  - Because they solve different problems. OpenAPI is great for describing endpoints, but devs still need conceptual docs, with things such as migration reasoning, caveats, examples, workflows, etc. AI systems need both.
- **Isn’t the very idea of Web MCP a security issue?**
  - Web MCP is gated by a tools Permissions Policy (defaults to self) – cross-origin iframes can’t register tools without explicit permission.

Join the Shift WhatsApp Group!

- Bringing news from https://shiftmag.dev/ directly to you – via WhatsApp

- Scan the QR code to join

Thank you

Edvin Teskeredzic

SENIOR AI SOFTWARE ENGINEER @ INFOBIP

Edvin.Teskeredzic@infobip.com

Sources

1. Stack Overflow 2025 Developer Survey — AI https://survey.stackoverflow.co/2025/ai/
2. Anthropic — Contextual Retrieval https://www.anthropic.com/news/contextual-retrieval
3. Anthropic — Effective Context Engineering for AI Agents https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents
4. Lost in the Middle: How Language Models Use Long Contexts (arXiv) https://arxiv.org/abs/2307.03172
5. Google — Optimizing for Generative AI Features on Search https://developers.google.com/search/docs/fundamentals/ai-optimization-guide
6. Nielsen Norman Group — Progressive Disclosure https://www.nngroup.com/articles/progressive-disclosure/
7. Write the Docs Newsletter — June 2026 (Shadow Docs) https://www.writethedocs.org/blog/newsletter-june-2026/#dealing-with-shadow-docs
8. Search Engine Journal — The Accessibility Tree Is How AI Agents Read Your Site https://www.searchenginejournal.com/the-accessibility-tree-is-how-ai-agents-read-your-site-its-breaking/578171/
9. web.dev — Build Agent-Friendly Websites https://web.dev/articles/ai-agent-site-ux
10. Chrome Blog — 15 Updates from Google I/O 2026: Powering the Agentic Web https://developer.chrome.com/blog/chrome-at-io26
11. llmstxt.org — The /llms.txt Specification https://llmstxt.org/
12. Infobip OpenAPI MCP — Expose OpenAPI-documented APIs as MCP servers https://github.com/infobip/infobip-openapi-mcp
13. arXiv — Build the Web for Agents, Not Agents for the Web https://arxiv.org/pdf/2506.10953
14. Chrome for Developers — WebMCP https://developer.chrome.com/docs/ai/webmcp
15. Chrome for Developers — WebMCP Declarative API https://developer.chrome.com/docs/ai/webmcp/declarative-api
16. How to Create AI-Ready and Human-Friendly Documentation with Contextual Density Mapping — https://www.infobip.com/developers/blog/how-to-create-ai-ready-and-human-friendly-documentation-with-contextual-density-mapping