DOCUMENTATION DOCTRINE

The non-negotiable rules that make UDA systems work.

This is not guidance. These are hard constraints. Violating them causes the system to degrade.

TABLE OF CONTENTS

- DOCUMENTATION DOCTRINE
  - TABLE OF CONTENTS
  - CORE PRINCIPLES
    - 1. One Page = One Answer to One Intent
    - 2. Strict Routing Discipline (Diataxis)
    - 3. No Mixing Modes
    - 4. No Narrative, No History
    - 5. No "Miscellaneous" or Catch-All Sections
  - TEMPLATE RULES
    - 6. Every Page Uses a Template
    - 7. Metadata Is Mandatory
    - 8. Cross-Linking Follows Routing Rules
  - MAINTENANCE RULES
    - 9. Event-Driven Updates (Not Calendar-Based)
    - 10. Ownership Is Metadata
    - 11. Half-Life Driven Review
  - CONTENT RULES
    - 12. Accessibility Rules
    - 13. Verifiability
    - 14. No Duplication
  - STRUCTURAL RULES
    - 15. Folder Structure Is Standardized
    - 16. Each Folder Has a README
    - 17. Templates Folder Structure
  - VIOLATION CONSEQUENCES
  - HOW TO ENFORCE
  - FINAL RULE

CORE PRINCIPLES

1. One Page = One Answer to One Intent

Rule: Every page must answer exactly one user question. If a page answers two questions, split it.

Why:

- Users search for specific answers, not broad topics
- Allows linking without ambiguity
- Makes updates surgical (change one thing, not five)
- Enables reuse across different contexts

Violation signs:

- Page is longer than 5 minutes to read
- Page covers multiple "how do I..." questions
- Title has "and" in it ("Concepts and Tasks")
- You can't write a one-sentence page summary

How to fix: Split into separate pages. Link them together.

---

2. Strict Routing Discipline (Diataxis)

Rule: Route each page to exactly one section based on user intent.

The Matrix (non-negotiable):

User Intent	Section	Page Type
"How do I...?"	Tasks	Task
"What is...?"	Concepts	Concept
"What are exact fields/flags?"	Reference	Reference
"How do I fix...?"	Troubleshooting	Troubleshooting
"How do I operate/maintain?"	Platform Ops	Runbook
"What changed?"	Release & Upgrade	Release Notes
"Why did we choose...?"	ADRs	ADR
"What does this mean?"	Glossary	Glossary

Why: Users develop mental models of where content lives. Violating the routing matrix breaks their mental model.

Violation signs:

- A page that explains a concept AND shows how to do a task
- Tasks that link to other Tasks
- Concepts that link to Tasks

- A Troubleshooting page that mixes diagnosis with conceptual explanation

How to fix: Delete explanatory text from Tasks. Delete procedural text from Concepts. Create separate pages.

---

3. No Mixing Modes

Rule: A page must be purely one mode: Task (procedural) OR Concept (explanatory) OR Reference (factual lookup).

Examples of violations:

Bad	Good
"How to configure authentication (and here's what authentication is...)"	Split into two pages: Task: Configure authentication and Concept: Authentication
"The Deployment API (which is used to deploy pods)"	Reference: Deployment API links to Task: Deploy an application, which links to Concept: Deployments
"Troubleshooting Pod failures (here's how Pod scheduling works)"	Troubleshooting: Pod won't start links to Concept: Pod scheduling

Why: Mixing modes overloads cognitive load. Users come for one thing; don't give them three.

---

4. No Narrative, No History

Rule: Pages must be timeless and concise. No "originally this was different" or "we used to do this but now we do that."

Violation examples:

- "Originally Deployments didn't support..."
- "In version 1.0 we had to..."
- "The history of X is..."
- "We chose this approach because..."

Why: Historical narratives add cognitive load without helping the user accomplish their goal right now.

How to fix:

- History → ADR: Why we chose X
- Evolution → Release Notes: v2.0 breaking changes
- Context → Concept: Background theory

---

5. No "Miscellaneous" or Catch-All Sections

Rule: If a page doesn't fit a template, split it until each piece fits exactly one template.

Violation examples:

- A page titled "Other considerations..."
- A page titled "Miscellaneous tips and tricks"
- A page with multiple unrelated "how do I" questions

Why: Catch-alls become unmaintainable dumping grounds.

How to fix: Create separate pages for each distinct piece.

TEMPLATE RULES

6. Every Page Uses a Template

Rule: No exceptions. Every page must use one of the 8 canonical templates (Task, Concept, Reference, Troubleshooting, Runbook, Release Notes, ADR, Glossary).

Why:

- Templates enforce consistency
- Lower barriers to contribution
- Enable validation and automation
- Make pages predictable for readers

Allowed customization:

- Section heading names (customize for your domain)
- Number of examples (minimum 1, no maximum)
- Metadata fields (add domain-specific fields)

Not allowed:

- Adding new top-level sections to a template
- Reordering template sections
- Removing core sections
- Creating a new template type

---

7. Metadata Is Mandatory

Rule: Every page must include frontmatter metadata with:

- `author` or `team`
- `last_verified` (YYYY-MM-DD)
- `status` (one of: current, needs_review, stale, superseded)

Domain-specific metadata varies by page type:

- Concepts: `audience` (Beginner/Intermediate/Advanced)
- Troubleshooting: `symptom_keywords` and `severity`
- Tasks: `prerequisites` in frontmatter
- Reference: `version_min` and `version_max`

Why: Metadata enables:

- Auto-flagging stale content
- Assigning ownership
- Filtering by audience
- Tracking verification dates

---

8. Cross-Linking Follows Routing Rules

Rule: Links between pages must follow Diataxis routing discipline:

- **Tasks** → Reference, Concepts, Troubleshooting (never Tasks)
- **Concepts** → Concepts, Reference (never Tasks)
- **Reference** → Concepts (never Tasks)
- **Troubleshooting** → Tasks, Concepts, Reference
- **Runbooks** → Reference, Concepts, Troubleshooting
- **Release Notes** → ADRs, Concepts
- **ADRs** → ADRs, Release Notes, Concepts
- **Glossary** → Concepts

Why: Prevents circular dependencies and maintains mental model clarity.

MAINTENANCE RULES

9. Event-Driven Updates (Not Calendar-Based)

Rule: Documentation updates are triggered by events, not arbitrary calendar reviews.

Events that trigger updates:

- New release/version: Update Reference, Release Notes, Tasks

- API deprecation: Flag all pages mentioning it
- Feature removal: Mark old page stale; link to replacement
- CVE: Flag Troubleshooting, Runbooks, Security
- Incident: Update Troubleshooting same day
- Architecture change: Create ADR; update related pages
- User feedback: Review and update page

Why: Calendar reviews create false confidence. Event-driven updates stay aligned with reality.

---

10. Ownership Is Metadata

Rule: Each top-level section has one owner. Each page has one steward. Store in metadata.

Ownership metadata:

```
section_owner: [Team Name]
page_steward: [Person Name]
review_cadence: [frequency, e.g., "monthly", "after release"]
```

Why:

- Accountability (not tribal knowledge)
- Enables tracking who to contact
- Clarifies who reviews/approves changes
- Makes delegation explicit

---

11. Half-Life Driven Review

Rule: Each doc type has an expected half-life. Auto-flag when threshold is exceeded.

Doc Type	Half-Life	Threshold	Action
Reference	6 months	Flag if > 6 months old	Verify current
Tasks	9-12 months	Flag if > 12 months old	Test end-to-end
Troubleshooting	12-18 months	Flag if > 18 months old	Verify still relevant
Runbooks	6-12 months	Flag after release	Test on current system
Release Notes	3-6 months	Auto-flag on next release	Update with new info
Concepts	18-24 months	Flag if > 24 months old	Verify accuracy
Tutorials	12 months	Flag if > 12 months old	Walk through
ADRs	Permanent	Never auto-flag	Only mark superseded

Why: Predictable half-lives catch rot before users notice.

CONTENT RULES

12. Accessibility Rules

Rule: All pages must be readable and searchable.

Required:

- Clear, unambiguous titles (search-friendly)
- No unexplained jargon (link to Glossary term or Concept)
- No images as the only explanation (images + text)
- Code examples properly formatted (syntax highlighting)

Why: Documentation is for everyone, not just experts.

---

13. Verifiability

Rule: Every fact in a page must be verifiable against current system state.

What this means:

- API field names match real API (not approximations)
- Command syntax works in current version
- Error messages match actual output
- Examples are runnable

Why: Incorrect documentation is worse than no documentation. Verify every claim.

---

14. No Duplication

Rule: If the same information appears in two pages, it's a bug.

Solution:

- Write the source of truth once
- Link to it from other pages
- If linking isn't possible, it's a routing problem (split the page)

Why: Duplicates become inconsistent over time.

STRUCTURAL RULES

15. Folder Structure Is Standardized

Rule: Use this exact folder structure:

```
your-kb/
├── 00-Start/
├── 10-Tutorials/
├── 20-Tasks/
├── 30-Concepts/
├── 40-Reference/
├── 50-Troubleshooting/
├── 60-PlatformOps/
├── 70-Security/
├── 80-ReleaseUpgrade/
├── 90-ADRs/
├── 99-Glossary/
└── README.md
```

Why: Numeric prefixes ensure consistent ordering. Users learn the structure once.

Exception: If you have fewer than 11 sections (e.g., an API KB with 7 sections), keep the numeric structure but skip unused ranges.

---

16. Each Folder Has a README

Rule: Every top-level folder has a README.md explaining:

- What this section is for
- When to use it
- When to use OTHER sections instead
- Key structural principles

Why: Reduces confusion. Users understand scope boundaries.

---

17. Templates Folder Structure

Rule: All templates live in `00-Start/Templates/`:

```
00-Start/Templates/
├── Template - Task.md
├── Template - Concept.md
├── Template - Reference.md
```

```
├── Template - Troubleshooting.md
├── Template - Runbook.md
├── Template - Release Notes.md
├── Template - ADR.md
└── Template - Tutorial.md
```

Why: Single source of truth for templates. Easy to find and update.

VIOLATION CONSEQUENCES

If you violate these rules:

1. **One Page = One Intent:** Pages become hard to link, hard to update, hard to maintain
2. **Routing Discipline:** Users get lost. Mental model breaks.
3. **No Mixing Modes:** Cognitive overload. Reduced readability.
4. **No Narrative:** Pages become bloated. Maintenance burden grows.
5. **No Catch-Alls:** Unmaintainable dumping grounds.
6. **Templates:** Inconsistency. Harder to contribute.
7. **Metadata:** No ownership. No auto-flagging. Docs decay silently.
8. **Routing Links:** Circular dependencies. Confusing navigation.
9. **Calendar Reviews:** False confidence. Docs stay wrong.
10. **No Ownership:** Unclear accountability. Nothing gets updated.
11. **No Half-Lives:** Rot goes undetected until it's too late.
12. **Inaccessibility:** Docs only work for experts.
13. **Unverifiable:** Docs become fiction.
14. **Duplication:** Inconsistency. Divergent versions of truth.
15. **Non-Standard Structure:** Users confused. Structure learning fails.
16. **No Folder READMEs:** Users don't know where to go.
17. **Scattered Templates:** Inconsistency. Hard to maintain.

HOW TO ENFORCE

Make these checks part of your workflow:

- **Pre-commit:** Template validation (correct structure, required sections)
- **PR review:** Routing audit (is this in the right section?)
- **Merge:** Metadata validation (author, last_verified, status present?)
- **Weekly:** Stale content check (auto-flag pages exceeding half-life)
- **Monthly:** Ownership audit (are owners still valid?)

FINAL RULE

If you break the rules, you break the system. UDA works because it's rigid, not flexible. Every violation is a choice to make documentation less maintainable.

There are good reasons for every rule. If you want to violate one, document why in an ADR first.

This is the doctrine. Follow it.