

THE

FOUNDRY OWNER'S MANUAL

How to Use Your AI Knowledge and Execution Hub

FORGED INTEGRITY

2026 EDITION

THE FOUNDRY OWNER'S MANUAL

How to Use Your AI Knowledge and Sovereign Execution Hub

Publisher: Forged Integrity Systems

Sovereign Founder: Christopher Lane

Canonical Platform: Foundry & Titus OS

Release ID: 2026.10.05-rev3

Protocol Revision: rev-2026.10.05.3

Source Digest:
`sha256_a24bdec3caf3609663e75c5d0b619000a9cdd32c01efe1f032c58cb83827b5aa`

Environment: PREVIEW (foundry-preview-dkv.pages.dev)

Deployed Commit: `d5ea396` (Cloudflare Pages)

Canonical URL: https://foundry-preview-dkv.pages.dev

Foreword & Core Operating Philosophy

"One brain. One source of truth. One command center. Many capture sources. Many AI models."

Foundry is the sovereign internal operating system and executive command center for Forged Integrity. It permanently replaces scattered documentation, stale prompts, and tribal knowledge with a deterministic, cryptographically auditable system that any human executive or authorized AI agent can pick up instantly without reconstructing context from scratch.

Table of Contents

Chapter 1: Quick-Start Guide (Founder Operational Flow) 1

1.1 Step 1: Open Foundry & Verify Founder Session

1.2 Step 2: Access Protocol Bootstrap (/start)

1.3 Step 3: Stage and Authorize Task Delegation (/tasks)

1.4 Step 4: Dispatch Instruction to Antigravity

1.5 Step 5: Verify Live Execution Receipts

1.6 Step 6: Review & Save Session (Canonical State Persistence)

Chapter 2: Navigation Surfaces & User Interface Map 4

2.1 Complete 14-Surface System Inventory

2.2 Illustrated Interface Layouts with Numbered Callouts

Chapter 3: Canonical Protocols & Foundational Constitutions 8

3.1 Verbatim Source Text vs Runtime Guidance

3.2 Mission & Vision (evi_titus_2df6720945462b58)

3.3 Titus Engineering Constitution (evi_titus_7e4b49bfb34e75d9)

3.4 Ethics & Principles Constitution (evi_titus_e9b46e3a6c9d09c6)

3.5 Session Startup & Shutdown Protocols

Chapter 4: Public Protocols vs Authenticated Private Context 12

4.1 Public Protocol Edge Distribution (/api/protocols)

4.2 Authenticated Private Gateway Context (/api/gateway/bootstrap)

Chapter 5: Authenticated Governance & Tasks Workbench 15

5.1 Task Envelope Definition & Ed25519 Signing

5.2 Truthful Receipt Automation: PreToolUse vs Post-Execution

5.3 The Agent-Directed Reporting Reality

Chapter 6: Delegated Execution Authority & Approval Fatigue Resolution 19

6.1 The One-Authorization Model

6.2 Root Cause Taxonomy of Historical Prompting

6.3 Measured Conditional Approval Accounting

6.4 Titus DEA Governance vs Antigravity Host Sandbox

Chapter 7: Security, Cryptography & Credentials 23

7.1 Asymmetric Authority & Key Separation

7.2 Protected PBKDF2 Password Setup & Tokens

Chapter 8: Worked Examples & Numbered Walkthroughs 26

8.1 Example A: Bounded Code Modification & Local Test

8.2 Example B: Context Retrieval & Brain Discovery

8.3 Example C: Proposal Staging & Conflict Resolution

8.4 Example D: Authorized Session Review, Approval & Atomic Persistence

Chapter 9: Known System Limitations & Boundaries 30

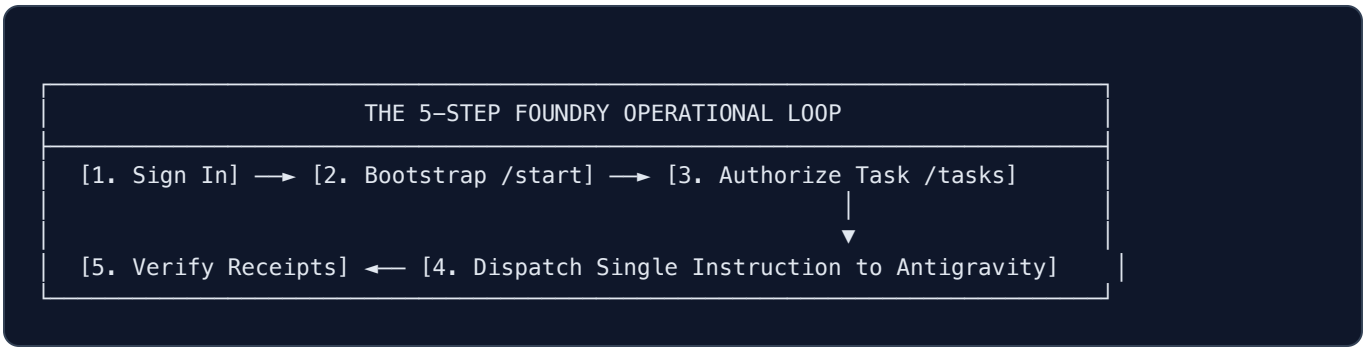
Chapter 10: Troubleshooting Runbook & Error Reference 32

Chapter 11: Technical Appendix & Glossary of Terms 35

Chapter 1: Quick-Start Guide

The Sovereign Founder Operational Lifecycle

Foundry is built around a deterministic, five-step loop that allows Founder Christopher Lane to delegate bounded authority to AI agents while retaining full sovereign governance and mathematical receipt verification.



Step 1: Open Foundry & Check Session

Navigate to the canonical production URL: `https://foundry-preview-dkv.pages.dev`. Inspect the top-right header:

- **Authenticated:** Displays `CHRISTOPHER • FOUNDER VERIFIED`. You possess full authority to stage and sign delegation grants.
- **Unauthenticated:** Displays a "Sign In" button. Click it to enter your Founder password. Session tokens are issued as secure HTTP-only cookies with PBKDF2 verification.

Step 2: Access Protocol Bootstrap (`/start`)

Click **"Start Here (AI)"** in the left sidebar or visit `/start`. This page provides the foundational starting context for any AI session:

- Review the Session Startup Protocol and Engineering Constitutions.
- Click **"Copy AI Bootstrap Prompt"** to copy a sanitized, deterministic context bundle into your clipboard.

Step 3: Stage and Authorize a Task (`/tasks`)

Navigate to the Tasks workbench at `/tasks`:

1. Enter the **Task Title** (e.g. `tmp/foundry-handoff-test.txt` or `docs/V1_OPERATIONAL_GUIDE.md`).
2. Configure the **Bounded Operational Envelope**:
    - **Allowed Actions:** `write_to_file`, `replace_file_content`, `run_command`, `view_file`.

- **Allowed Target Paths:** `docs/`, `lib/`, `tmp/`, `scratch/`.
- **Execution Budget:** e.g., `10` maximum executions.
- **Risk Ceiling:** `R2_INTERNAL_GOVERNED`.

3. Click **"Authorize Task Delegation"**. Foundry's server uses the private key to issue a signed Ed25519 grant (`del_grant_*`).

Step 4: Dispatch to Antigravity

Foundry immediately displays the formatted, copyable **Start in Antigravity** instruction:

```
Execute assigned Foundry task "tmp/foundry-handoff-test.txt" under bounded delegation del_grant_1
```

Paste this single instruction into your Antigravity pairing session. The agent retrieves its authorized scope and proceeds autonomously without asking redundant questions.

Step 5: Verify Live Execution & Close Grant

Return to `/tasks` in your browser and click **"Refresh State"**. The live display updates directly from Neon PostgreSQL:

- **Refreshed Timestamp:** Shows live UTC confirmation.
- **Grant Status:** Shows `COMPLETED` and execution usage (e.g., `2/10 Executions`).
- **Receipts Table:** Lists every executed host tool call with its unique `cxr_*` ID, timestamp, and status.

Step 6: Review & Save Session (Canonical State Persistence)

In the Command Composer (`/`), click **"Review & Save Session"**:

1. **Inspect Proposed Changes:** View exact diffs for the Builder Journal, Current State version progression, and optional EAP updates in the modal.
2. **Verify Server Signature:** Confirm the `FOUNDER VERIFIED: CHRISTOPHER LANE` badge certifying HMAC draft integrity.
3. **Approve & Commit:** Click **"Approve & Save Session"**. The server commits records atomically to Neon PostgreSQL in a single CTE transaction.
4. **Fresh Retrieval:** A fresh conversation immediately retrieves the persisted session records via the normal AI provider.

Chapter 2: Navigation Surfaces & UI Map

The 14 Supported Sovereign Workspaces

Foundry organizes institutional intelligence and executive operations into fourteen primary navigation surfaces, accessible via the persistent left sidebar and universal search modal (`Cmd+K` / `Ctrl+K`).

FOUNDRY COMMAND SHELL • INTERFACE MAP — 14 NAVIGATION SURFACES

1. **/ (Command Center):** Executive pulse, Core lifecycle state machine, real-time telemetry.
2. **/#today (Today):** Daily priorities, pending Founder review queue, and active timers.
3. **/start (Start Here AI):** Unauthenticated canonical protocol bootstrap and AI prompts.
4. **/ask (Ask Foundry):** Dynamic intelligence workspace with conversation trees.
5. **/projects (Projects):** Multi-project registry, active EAP milestones, repository links.
6. **/products (Products):** Product specifications, versioned artifacts, packaging status.
7. **/people (People):** Executive team hierarchy, domain authority maps, agent principals.
8. **/decisions (Decisions):** Permanent Decision Log with rationale and evidence citations.
9. **/tasks (Tasks & Delegations):** Bounded delegation workbench and live receipt monitor.
10. **/sources (Sources):** Evidence capture repository, Notion/GitHub sync status.
11. **/review (Review Queue):** Candidate knowledge staging, conflict resolution interface.
12. **/history (History):** Immutable audit trail of action executions and Founder gates.
13. **/system (System):** Environment health diagnostics, database latency, API status.
14. **/help (Help & Manual):** In-app Owner's Manual with PDF and EPUB e-book downloads.

Detailed Surface Analysis

1. Command Center (`/`) & Today's Pulse (`/#today`)

The primary dashboard for Founder Christopher Lane. Features the living Core Lifecycle state machine (`UNDERSTAND` , `RETRIEVE` , `REASON` , `ACT`), token telemetry counters, active project milestones, and pending approval alerts.

2. Start Here (`/start`) — The AI Bootstrap

The universal starting point for any human or AI contributor. Exposes public constitutions, startup protocols, and provides a one-click **"Copy AI Bootstrap Prompt"** button that compiles all essential rules into a sanitized prompt.

3. Tasks & Delegations (`/tasks`)

The governance execution cockpit. Allows the Founder to define bounded authority envelopes (allowed tools, target directories, execution budgets), signs grants using asymmetric Ed25519 cryptography, and polls live execution receipts (`cxr_*`) from PostgreSQL.

4. Review Queue (`/review`) & Conflict Resolution

Guards canonical truth. Whenever raw evidence contradicts existing knowledge, a conflict item is staged in the Review Queue. The Founder can inspect conflicting evidence diffs, approve canonical updates, or reject unverified claims.

5. Help Center & Manual (`/help`)

The in-app documentation center. Features comprehensive guides on quick-start workflows, DEA risk classes, verbatim constitutions, troubleshooting runbooks, and direct download buttons for this Owner's Manual in PDF, EPUB, and Markdown formats.

Chapter 3: Canonical Protocols & Constitutions

Verbatim Source Records vs Runtime Guidance

A central tenet of Foundry is the strict separation between **verbatim imported source text** (derived from Titus consolidation records stored in Neon PostgreSQL under SHA-256 evidence integrity) and **runtime operational guidance** (which translates those rules into executable tool checks).

> **Evidence Integrity Rule:** The canonical text of constitutions and checklists must never be paraphrased, truncated, or altered. Runtime engines reference the raw evidence ID directly.

1. Forged Integrity Mission & Vision

Entity ID: `ent_gov_mission` • **Evidence ID:** `evi_titus_2df6720945462b58`

> ***Mission:*** *Give every Forged Integrity organization and project a durable, AI-native home — so starting or resuming work never requires reconstructing context from scratch.*
>
> ***Vision:*** *Titus becomes Forged Integrity's executive operating system: a universal platform with personal workspaces where current state, decisions, knowledge, execution, and handoffs live together.*
>
> ***Purpose:*** *Replace scattered docs, stale prompts, and tribal knowledge with one deterministic, local-first system that any AI — or any person — can pick up instantly.*

2. Titus Engineering Constitution

Entity ID: `ent_gov_engineering` • **Evidence ID:** `evi_titus_7e4b49bfb34e75d9`

1. **Current State is the single source of truth.** Never duplicate project identity, version, milestone, specification, current task, or next task anywhere else.
2. **Every workspace is isolated.** No project's data leaks into another's Knowledge Engine.
3. **All persistence is local** until a backend milestone explicitly says otherwise.
4. **Every architectural decision gets logged** in the Decision Log with a rationale — not just the what, but the why.
5. **Specifications precede implementation** for anything non-trivial.
6. **The Builder Journal is a running log,** not a changelog — write it in the moment, not after the fact.
7. **AI context packages are deterministic.** No hidden state, no AI API calls to generate them.
8. **Ship the smallest working version** of a milestone, then iterate.

3. Titus Ethics & Principles Constitution

Entity ID: `ent_gov_ethics` • **Evidence ID:** `evi_titus_467af376bdd315fb`

1. **Protect Christopher and sovereign human agency in all operations.**
2. **Maintain truthfulness and evidence traceability for every claim.**
3. **Keep boundaries strict between projects, workspaces, and external integrations.**
4. **Execute high-impact operations only under explicit, verified authorization.**
5. **Provide auditable receipts and transparent decision rationale for all actions.**
6. **Preserve continuity across sessions through structured state and handoffs.**

4. Session Startup & Shutdown Protocols

Startup Protocol (`ent_gov_startup_protocol` • Evidence: `evi_titus_112a0d0b0d2d5a64`):

1. **Step 1: Load Workspace** — Open project workspace root and resolve repository structure.
2. **Step 2: Read Current State** — Inspect project identity, version, milestone, and active task.
3. **Step 3: Read Builder Journal** — Review recent work log entries to avoid repeating past attempts.
4. **Step 4: Read Decisions** — Inspect architectural decision records to preserve design constraints.
5. **Step 5: Read Active EAP** — Review executive action plan and immediate objectives.
6. **Step 6: Generate AI Context** — Assemble operational envelope and verify proposed tool calls remain within the granted risk ceiling.

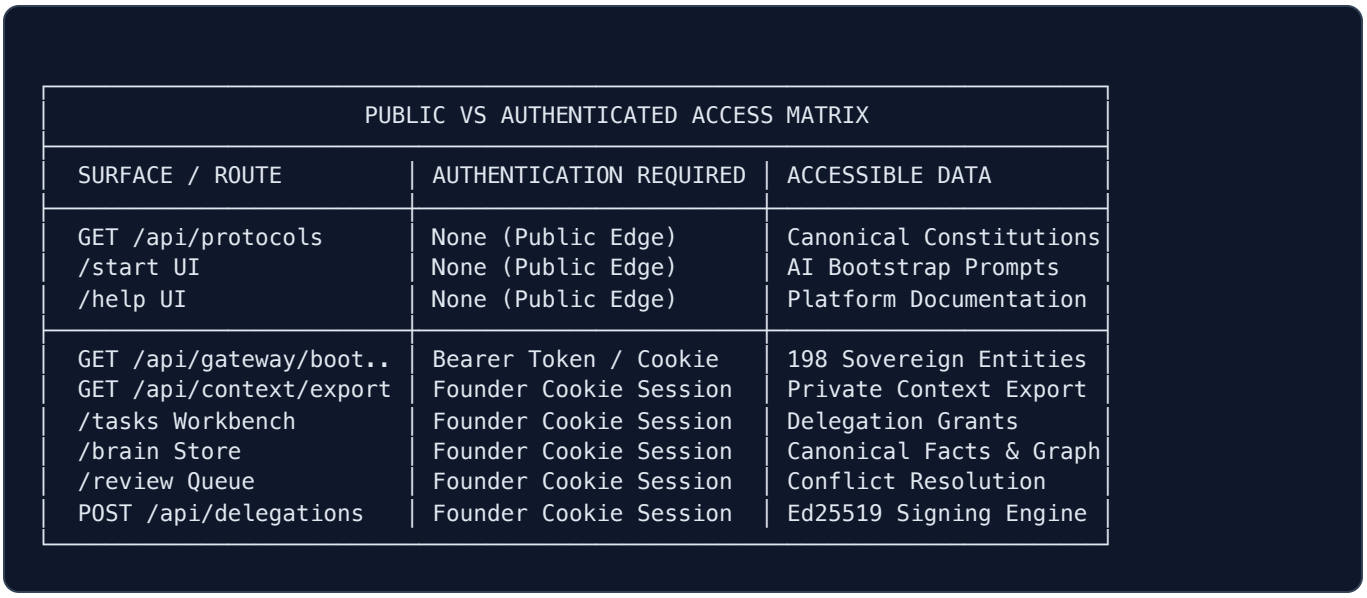
Shutdown Protocol (`ent_gov_shutdown_protocol` • Evidence: `evi_titus_1a66d2860d05e80d`):

1. **Step 1: Update Builder Journal** — Record completed work, test outcomes, and discoveries in the running log.
2. **Step 2: Update Decisions** — Document any new architectural decisions with rationale and consequences.
3. **Step 3: Update Current State** — Refresh current state with updated version, completed milestones, and next task.
4. **Step 4: Update EAP** — Advance executive action plan items to reflect finished steps.
5. **Step 5: Generate Context Package** — Confirm execution receipts posted to `/api/connectors/receipts` and close grant via `/api/delegations/{id}/complete`.

Chapter 4: Public vs Private Context

Sovereign Architecture & Data Boundaries

Foundry implements a strict, architectural separation between open protocol bootstrapping, protected internal company context, and bounded tool execution.



1. The Access Paths and Authority Triad

1. **Public Protocol Reading (`/start`, `/api/protocols`):** Zero authentication required. Returns foundational constitutions, checklists, and DEA risk tiers. Grants **ZERO** execution authority.
2. **Scoped Private Reading & Context Exports (`/api/gateway/bootstrap`, `/tasks`, `/api/context/export`):** Requires Founder authentication (session cookie or bearer secret). Returns private company facts, decisions, and uncommitted queues. Exporting private context requires an explicit Founder action.
3. **Bounded Tool Execution (`del_grant_*`):** Requires a cryptographically signed Ed25519 Delegation Grant. Inside the signed envelope, routine tool operations proceed autonomously without repetitive confirmations.

2. Client Support Matrix

CLIENT ENVIRONMENT	SUPPORT STATUS	INTEGRATION & VERIFICATION DETAILS
Antigravity (Google / IDE)	WORKING	Full integration: authenticated handoff, PreToolUse hook bridge (`scripts/dea-pretool-bridge.cjs`), automatic DEA risk evaluation, and agent-directed `cxr_*` receipt persistence to PostgreSQL.
Ordinary AI Chat (Web / Claude / ChatGPT)	SUPPORTED	Zero-credential protocol bootstrap via `/start` + Founder-authenticated context package export via `/api/context/export?project=foundry`. Operates in advisory/drafting mode: prepares *proposed* journal updates, decisions, and receipts as drafted text for Founder review without claiming unexecuted persistence or grant completion.
Claude Code (Anthropic CLI)	PROPOSED / UNVERIFIED	Integration architecture specified (PreToolUse / PostToolUse hook bindings). Labeled unverified until tested directly in a Claude Code environment.

3. Shared Missing-Context Checklist

When an AI session operates without authenticated gateway access, it must verify the following core items or import an authorized context package from `/api/context/export`:

1. **Workspace Root & Directory Layout:** Target repository structure and absolute file paths.
2. **Target Project & Canonical ID:** Canonical project identifier (e.g., `foundry`, `titus-core`).
3. **Active Git Branch & Remote:** Current working branch and upstream repository status.
4. **Current Milestone & Immediate Goal:** Active sprint deliverable or specific task objective.
5. **Active Decision Log & ADRs:** Binding architectural constraints.
6. **Unresolved Conflicts & Review Items:** Uncommitted candidate evidence.
7. **Concurrent-Work Awareness:** Identify other active agents or sessions operating in the same repository or target paths before making changes; if concurrent state is unknown, report that uncertainty explicitly rather than assuming sole ownership or inventing a claims registry.
8. **Signed Delegation Grant:** Active `del_grant_*` envelope if tool execution is required.
9. **Authorized Action Envelope:** Allowed tool names (`run_command`, `write_to_file`, etc.).
10. **Authorized Target Directory Paths:** Scoped directory prefixes (`docs/`, `lib/`, `app/`).
11. **Maximum Execution Budget & Escalation Criteria:** Hard ceiling on tool calls and emergency escalation contact.

Client-Specific Additions

- **Antigravity:** Local bridge verification (`scripts/dea-pretool-bridge.cjs`), active delegation cache (`tmp/active-delegation.json`), receipt persistence to `/api/connectors/receipts`.
- **Ordinary AI Chat:** Read-only advisory mode, Founder-exported context package via `/api/context/export`, proposed shutdown updates formatted as unpersisted drafts.
- **Claude Code (Unverified):** PreToolUse / PostToolUse hook bindings, unauthenticated public bootstrap via `/start`.

Chapter 5: Governance & Tasks Workbench

Task Envelopes, Ed25519 Signatures & Agent-Directed Receipts

The Tasks Workbench (`/tasks`) is the central execution hub where human authority is transformed into bounded, cryptographic machine grants.

1. The Anatomy of a Delegation Grant

When the Founder clicks **"Authorize Task Delegation"**, the server constructs an immutable grant envelope:

```
{
  "id": "del_grant_1791185935030_v8oc",
  "name": "tmp/foundry-handoff-test.txt",
  "allowedActions": ["write_to_file", "replace_file_content", "run_command", "view_file"],
  "allowedTargets": ["tmp/", "scratch/"],
  "riskCeiling": "R2_INTERNAL_GOVERNED",
  "startsAt": "2026-10-05T18:00:00.000Z",
  "expiresAt": "2026-10-06T18:00:00.000Z",
  "maxExecutions": 10,
  "executionCount": 0,
  "status": "ACTIVE",
  "signature": "3a8f9c2d1e... (Ed25519 signature)",
  "envelopeDigest": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
}
```

2. Honest Receipt Accounting: PreToolUse vs Post-Execution Proof

Foundry enforces a strict conceptual and operational distinction between authorization and proof of execution:

ATTRIBUTE	PRETOOLUSE HOOK (AUTHORIZATION GATE)	RECEIPT CAPTURE (PROOF OF WORK)
Substrate & Scope	**Antigravity-Specific** (`scripts/dea-pretool-bridge.cjs` hook).	`POST /api/connectors/receipts` → Neon PostgreSQL.
Timing	Evaluated *before* the tool call executes.	Recorded *after* the tool finishes on the host.
Identifier	Hook Evaluation ID / Step Index.	Unique Receipt ID (`cxr_1791186052320_8ywhh6s`).
Meaning	"The proposed tool call is allowed to run."	"The tool completed with status SUCCESS and exit code 0."

Substrate Reality & Agent Reporting: PreToolUse hook enforcement is Antigravity-specific. Ordinary AI chats operate behaviorally without local hook interception and draft proposed receipts for Founder review. Furthermore, authorization by a PreToolUse hook does *not* create a receipt; in agent workflows, the agent or client bridge explicitly posts completed receipts back to Foundry.

Chapter 6: DEA & Approval Fatigue Resolution

The One-Authorization Model & Measured Accounting

Historically, pairing sessions suffered from repetitive, friction-heavy confirmation prompts. We resolved approval fatigue by aligning the full execution envelope with Titus DEA governance and Antigravity host sandbox rules.

1. Root Causes of Approval Friction

1. **Unnecessary Agent Confirmation:** The agent asked conversational permission for operations already inside the active grant.
2. **Missing Local Grant Cache:** `tmp/active-delegation.json` was cleared after a single test script, causing subsequent edits to fall back to `ask`.
3. **Overly Narrow Target Envelopes:** Grants restricted solely to `tmp/` blocked authorized project updates (`docs/`, `lib/`, `app/`).
4. **Hook Classification Gaps:** Non-destructive supporting commands (`git status`, `npm test`, `tsc`) were not recognized as R0 read operations.
5. **Host Sandbox Network Boundaries:** Commands needing outbound network access (`git push`, Cloudflare `pages deploy`) triggered OS-level host dialogs.

2. The Three Executive Tiers & Domain Authorities

Foundry organizes governance across **3 Executive Tiers**, establishing clear separation between supreme sovereign authority, enterprise coordination, and bounded domain delivery:

Tier 0 — Sovereign Founder (Christopher Lane): Supreme authority, strategic direction, asymmetric cryptographic signing, and final approval on all destructive (R4) and enterprise actions.

Tier 1 — Enterprise Executive (Titus): Operational orchestration, multi-domain alignment, protocol enforcement, and cross-project handoff validation.

Tier 2 — Bounded Domain Executives (4 Specialized Agents):

- **Samson (Defense & Infrastructure):** System resilience, security audit, CI/CD pipeline integrity, and runtime boundary verification.
- **Joseph (Treasury & Resource Allocation):** Token budget governance, API consumption monitoring, and infrastructure cost accounting.
- **The Quill (Brand & Communications):** Documentation clarity, owner manuals, external reporting, and public protocol communications.
- **The Craftsman (Product & Engineering Delivery):** Code implementation, architectural design record compliance, unit testing, and artifact builds.

3. Authoritative DEA Action & Risk Matrix

RISK TIER	CANONICAL SCOPE	REPRESENTATIVE ACTIONS & COMMANDS	AUTHORITY & PROMPTS
R0_READ	Read-Only Inspection	`view_file`, `list_dir`, `git status`, `git log`, `git diff`, `GET /api/protocols`	`0 PROMPTS (AMBIENT / AUTO-ALLOWED)`
R1_INTERNAL_REVERSIBLE	Reversible Local Edits	`replace_file_content` (in-scope), `write_to_file` (`docs/`, `tmp/`)	`0 PROMPTS (UNDER ACTIVE GRANT)`
R2_INTERNAL_GOVERNED	Governed Local Mutations	`npm run build`, `npm test`, `tsc`, DB migrations, Review Queue staging	`0 PROMPTS (UNDER SIGNED GRANT)`
R3_EXTERNAL_MUTATION	External / Remote Changes	`git push`, Cloudflare `pages deploy`, SaaS API mutations (secret rotation is Founder-reserved)	`FOUNDER SIGNED GRANT + HOST DIALOG`
R4_HIGH_IMPACT	Destructive Operations	Database drop/truncate, destructive credential deletion, domain transfers	`PROHIBITED / MANUAL ONLY`

4. Precision of requiresSignature vs Delegation Scope

A risk tier's signature requirement (`requiresSignature: true` for R2/R3) dictates whether the operation type requires an asymmetric cryptographic Founder signature on the grant envelope. R1 (`requiresSignature: false`) does **not** require a cryptographic signature at the policy level, but R1 is **not** broadly authorized by default. Under Zero Ambient Authority, mutating tool operations (such as file edits) still strictly require an active, in-scope delegation envelope (`allowedActions` and `allowedTargets`) or explicit Founder confirmation.

5. Measured Conditional Approval Accounting (Empirical Run: `del_grant_1791185935030_v8oc`)

Approval counts are conditional upon the specific tool/network action and grant envelope. In verified test run `del_grant_1791185935030_v8oc` (2/10 execution budget used):

LIFECYCLE PHASE	TOOL OPERATION	GOVERNING SUBSTRATE	APPROVAL PROMPTS	CONDITION / RATIONALE
1. Task Authorization	Founder signs grant on /tasks	Foundry Server	`1 HUMAN CLICK`	Sovereign Founder bounds scope and signs grant.
2. Context Discovery	`view_file`, `git status`	Titus DEA (R0)	`0 PROMPTS`	Read-only discovery auto-allowed under R0 policy.
3. Scoped Modifications	`replace_file_content`	Titus DEA (R1)	`0 PROMPTS`	Target file matches authorized grant envelope.
4. Local Build & Test	`npm run build`, `tsc`	Titus DEA (R2) / Sandbox	`0 PROMPTS`	Sandboxed local execution within budget.
5. Telemetry Receipts	`POST /api/connectors/receipts`	Titus / Foundry	`0 PROMPTS`	Receipt logging authorized autonomously.
6. Remote Network Deploy	`git push`, `pages deploy`	Antigravity Host Sandbox	`CONDITIONAL HOST DIALOG`	**Conditional:** Required only when outbound network access bypasses sandbox (`BypassSandbox: true`).
7. Task Completion	Final summary report	Antigravity / Founder	`0 PROMPTS`	Delivered directly upon task conclusion.

6. Task Authorization vs Host Sandbox Boundary

Crucial Distinction: Task authorization (Ed25519 Delegation Grant) and Host Sandbox confirmation (`BypassSandbox: true`) are two distinct security boundaries:

- A host dialog does *not* grant missing institutional Founder authority.
- A valid delegation grant does *not* eliminate host OS-level sandbox security boundaries for external network or unsandboxed binaries.

Chapter 7: Security & Cryptography

Asymmetric Authority & Zero Ambient Authority

Foundry enforces strict cryptographic separation to guarantee that AI agents and local execution daemons can never self-authorize, expand their permissions, or forge credentials.



1. Key Separation & Absence Verification

- **Private Signing Key:** Stored exclusively in Cloudflare Pages encrypted environment secrets. Used solely by server endpoints to sign grants.
- **Public Verification Key:** Stored in local configuration (`.env.local`) or process environment. Used by `scripts/dea-pretool-bridge.cjs` to verify signatures.
- **Verified Absence:** `FOUNDRY_PRIVATE_SIGNING_KEY` is strictly absent from local executor environments, ensuring total isolation between Founder authority and machine execution.

2. Protected Founder Authentication & Setup Tokens

- **PBKDF2 Password Hashing:** 100,000 iterations of SHA-256 with cryptographically unique salts.
- **Single-Use Setup Token:** First-time Founder password creation requires an owner-controlled high-entropy token (`FOUNDRY_SETUP_TOKEN_HASH`). The token is hashed, stored securely, and consumed atomically upon enrollment.
- **Session Tokens:** High-entropy signed session cookies (`foundry_founder_session`) with `HttpOnly`, `SameSite=Lax`, and `Secure` flags. Logout invalidates sessions immediately.

Chapter 8: Worked Examples

Step-by-Step Practical Scenarios

Scenario A: Bounded Code Refactoring & Unit Testing

Goal: Update `docs/V1_OPERATIONAL_GUIDE.md` and run `npm run build` without intermediate approval interruptions.

1. Founder signs in to `https://foundry-preview-dkv.pages.dev/tasks`.
2. Sets Task Title: `docs/V1_OPERATIONAL_GUIDE.md`.
3. Configures envelope: Allowed Actions (`replace_file_content`, `run_command`), Allowed Targets (`docs/`), Budget (`10`).
4. Copies the generated instruction into Antigravity.
5. Antigravity queries `/api/gateway/bootstrap`, executes `replace_file_content` (auto-allowed by PreToolUse), runs `npm run build` (auto-allowed by PreToolUse), and submits receipt `cxr_*` to `/api/connectors/receipts`.
6. Founder clicks **"Refresh State"** on `/tasks` and sees `COMPLETED (2/10)` with both verified receipts.

Scenario B: Context Retrieval & Sovereign Brain Query

Goal: Query institutional facts about executive domains without modifying database state.

1. Agent calls `view_file` or queries `GET /api/protocols`.
2. PreToolUse hook evaluates tool as `R0_READ` and immediately returns `allow` with zero prompts.
3. Agent synthesizes context and delivers answer to the Founder.

Scenario C: Candidate Knowledge Proposal & Conflict Review

Goal: Ingest candidate facts from Notion evidence and resolve contradictions.

1. Evidence ingestion identifies a contradiction between Notion raw text and Neon canonical facts.
2. Action Engine stages a review item in `/review` (`R2_INTERNAL_GOVERNED`).
3. Founder navigates to `/review`, inspects the evidence diff, and clicks **"Approve Update"** or **"Reject Claim"**.

Scenario D: Authorized Session Review, Approval & Atomic Persistence

Goal: Conclude an active work session, review drafted accomplishments and state changes, and atomically commit them to Neon PostgreSQL without terminal scripts.

1. Founder opens Command Composer (`/`) under authenticated session (`foundry_founder_session`).

2. Clicks **"Review & Save Session"**. Foundry calls `POST /api/session/draft` to synthesize proposed records and issue an HMAC-SHA256 signature.
3. Review modal opens with title *"Review & Authorize Session Save"* and badge *"Founder Verified: Christopher Lane"*.
4. Founder inspects proposed diffs: new Builder Journal entry and Current State progression (preserving `datedSnapshot.asOf` verbatim).
5. Founder clicks **"Approve & Save Session"**. Foundry executes `POST /api/session/save`, verifying Founder authority and cryptographic signature match.
6. Neon PostgreSQL transaction executes atomically (inserting the journal record and incrementing the Current State version in a single CTE query).
7. UI displays green confirmation banner with exact persisted IDs (e.g., `ent_asset_journal_*` and `ent_asset_current-state-* (v9)`).
8. Founder opens a fresh conversation and queries the saved journal entry. The normal AI provider retrieves and cites the record accurately.

Chapter 9: Limitations & Boundaries

Truthful Architectural Realities

To preserve sovereign integrity, Foundry V1 enforces strict, intentional system boundaries:

1. No Ambient Auto-Launch

Antigravity does not possess an external background daemon for unprompted, autonomous session creation. Christopher initiates the pairing session and pastes the single bootstrap instruction.

2. Cloudflare Edge Neon HTTPS Fetch Connectivity

In Cloudflare Pages Worker runtime, database queries execute over HTTPS fetch (`neonConfig.poolQueryViaFetch = true`) to avoid edge WebSocket origin DNS lookup issues. Direct TCP sockets are used only in local Node.js environments.

3. Receipt Submission Timing

Receipts reflect completed operations explicitly reported to `/api/connectors/receipts`. If an agent terminates unexpectedly mid-task before posting, unsubmitted receipts will not appear in the workbench.

4. Read-Only External SaaS Connectors

Notion and GitHub integrations in V1 are strictly read-only capture sources. Foundry ingests external data as candidate evidence; it does not perform autonomous external writes.

Chapter 10: Troubleshooting Runbook

Operational Error Reference & Resolutions

ERROR MESSAGE / SYMPTOM	ROOT CAUSE	IMMEDIATE STEP-BY-STEP RESOLUTION
"Unauthorized: Missing Founder authorization credentials"	Founder session cookie expired or unauthenticated request to `/api/delegations`.	Click **"Sign In as Founder"** on `/tasks` or `/start` and log in with your Founder password.
"Refresh State shows no change"	Receipts query filtered by connector ID or host tool call did not submit receipt payload.	Confirm task completed on host. Click **"Refresh State"**. The live timestamp updates immediately. Check that the grant ID in Antigravity matches the active grant.
"Action denied: outside delegated scope"	Requested tool call or target folder is not listed in the active delegation grant.	Stage a new grant on `/tasks` containing the required action (e.g. `run_command`) and target directory, then authorize.
"Single-use setup token invalid or expired"	Token was already consumed or exceeded its 24-hour lease.	Generate a fresh setup token locally via `node scripts/generate-setup-token.mjs`.
"401 Unauthorized on /api/gateway/bootstrap"	Anonymous caller attempted to access private project context.	Supply `Authorization: Bearer FOUNDRY_AGENT_SECRET` in request headers or sign in as Founder via browser.

Chapter 11: Technical Appendix & Glossary

1. Database Schema & Migration Sequence

Foundry persistence relies on Neon PostgreSQL with eight forward-only migrations in `db/migrations/`:

- `0001_initial_schema.sql` • Entities, Relationships, Knowledge Items, Evidence, Audit Log.
- `0002_evidence_r2_and_search_index.sql` • Full-text search indices and R2 object pointers.
- `0003_ai_workspace_conversations_artifacts_telemetry.sql` • Multi-turn conversations, messages, artifacts.
- `0004_phase4_intelligence_workspace.sql` • Intelligence Workspace extensions and prompt registry.
- `0005_phase5_governed_action.sql` • Action intents, Founder approvals, executions, automations.
- `0006_phase7_delegated_automation.sql` • Delegation grants and execution limits.
- `0007_phase8_bounded_agents.sql` • Bounded agent definitions and timeline step events.
- `0008_phase8_schema_alignment.sql` • Action engine ID widening (`VARCHAR(128)`).

2. Glossary of Terms

DEA (Delegated Execution Authority)
: Foundry's governance substrate classifying all operations into five risk tiers (R0 to R4) and enforcing cryptographic bounds.

R0_READ
: Read-only discovery and inspection operations. Auto-allowed without Founder signature.

R1_INTERNAL_REVERSIBLE
: Local file modifications and temporary drafts. Pre-authorized under active delegation grants.

R2_INTERNAL_GOVERNED
: Code modifications, builds, tests, and review queue staging. Requires explicit Founder signature.

R3_EXTERNAL_MUTATION
: Third-party SaaS state modifications. Requires explicit Founder signature.

R4_HIGH_IMPACT
: Destructive operations (database drops, DNS edits). Prohibited from automated execution.

PreToolUse Hook
: Lifecycle hook intercepting tool calls before execution to return allow, ask, or deny.

Execution Receipt (`cxr_*`)
: Cryptographically verifiable record of completed host tool execution stored in Neon PostgreSQL.