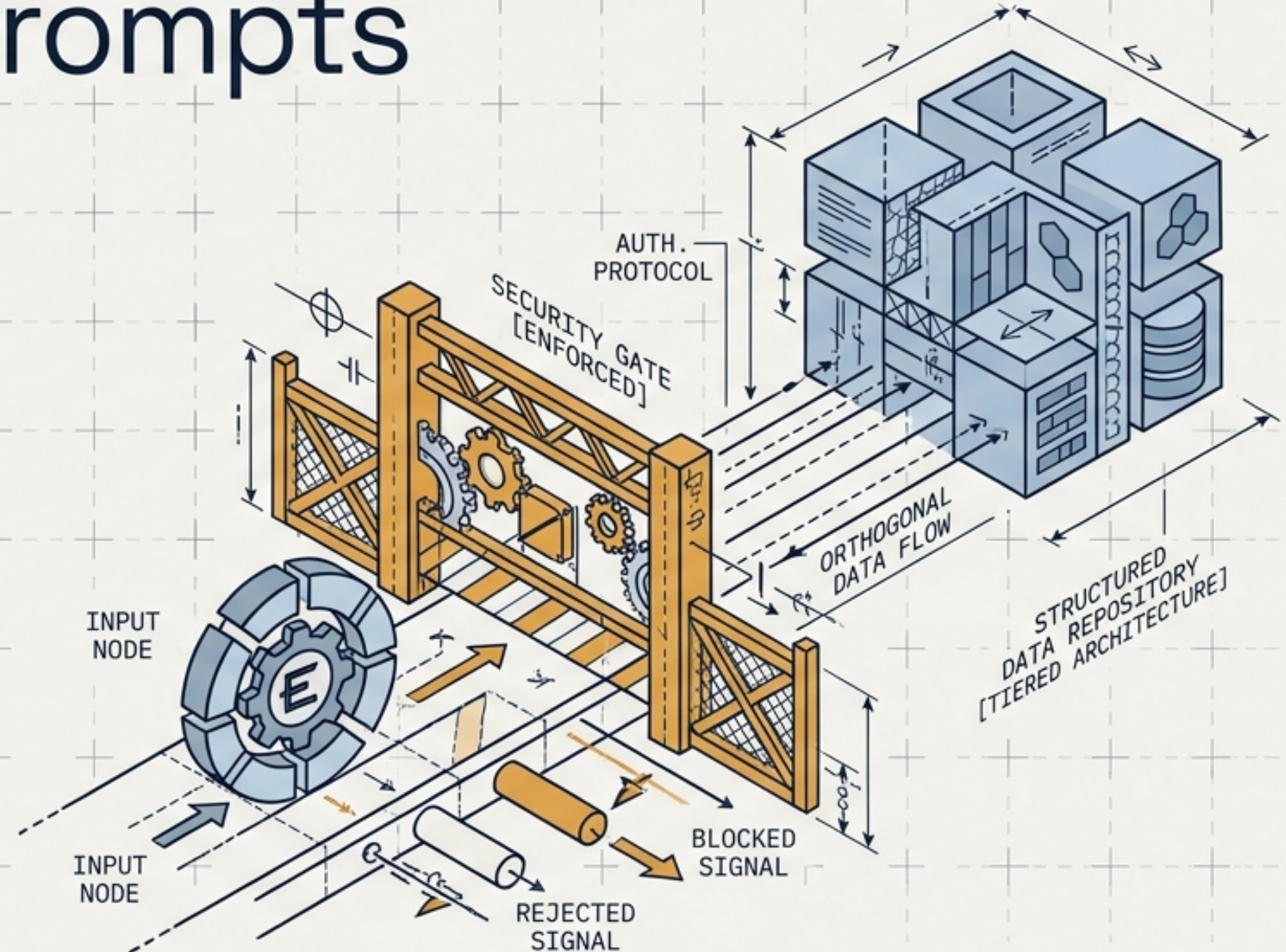


Moving from Prompts to Structural Enforcement

The Tiered Startup Architecture for AI Coding Agents.

[SYSTEM_INIT] [STATE: PENDING]



The Cost of Unmanaged AI Sessions

Context Waste



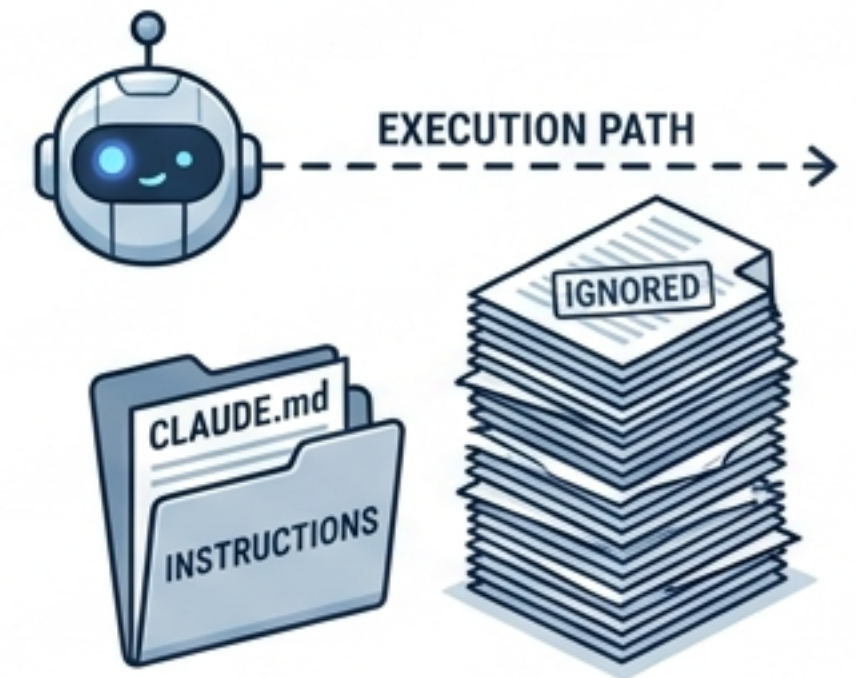
Loading a 3000-line rule set burns ~4K tokens every session—even for a simple typo fix.

Rule Drift



Instructions claim '58 rules', but the database has 62. Facts go stale until an error occurs.

Startup Chaos



Agents skip CLAUDE.md instructions, loading partial context before executing tasks.

Writing 'you must read X' in a markdown file is documentation. It is not enforcement.

Documentation vs. Enforcement

The Old Way

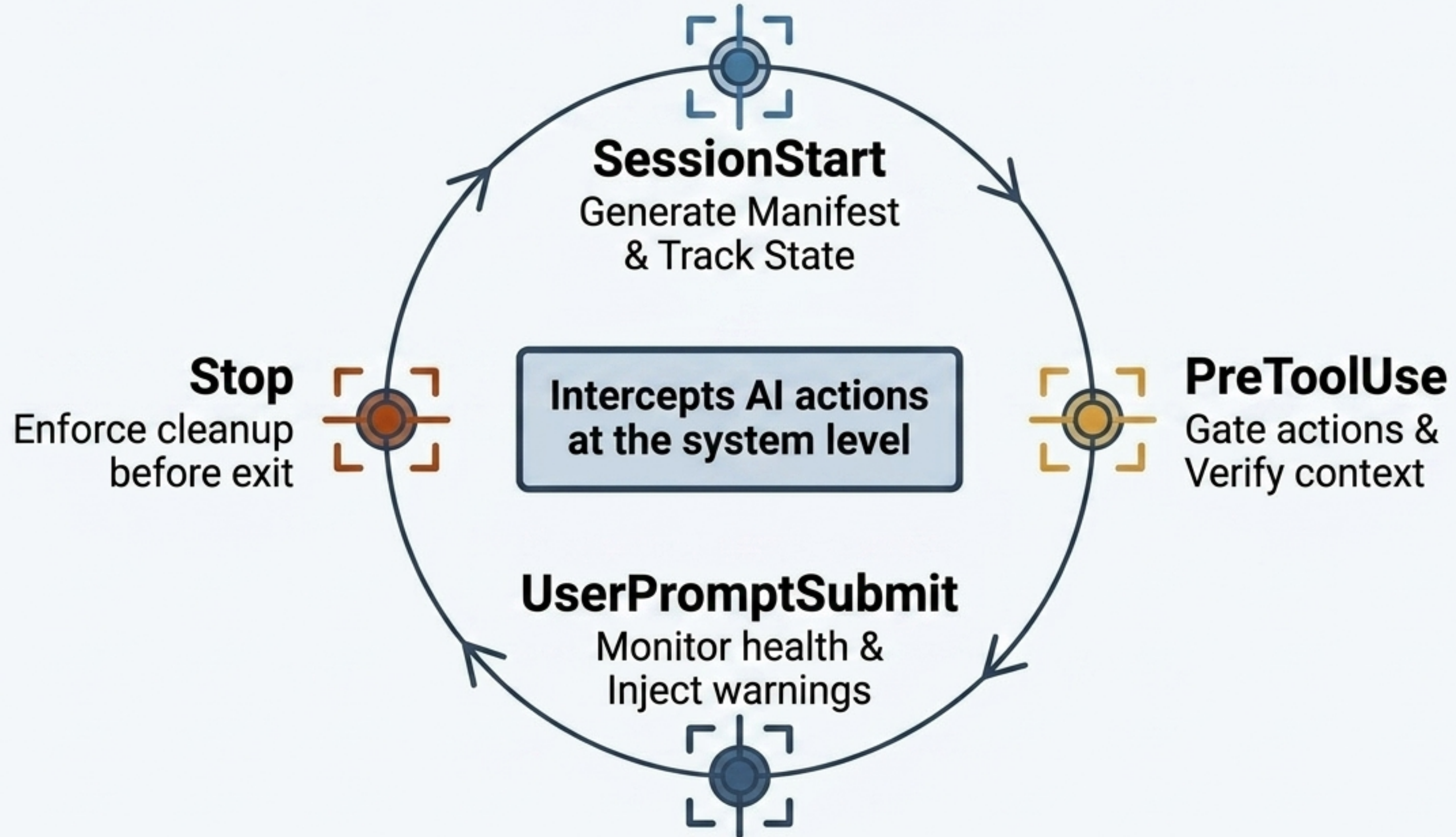
- Suggestion-based compliance
- Relies on LLM reading instructions voluntarily
- Trusts shell exit codes (\$?)
- Unbounded self-healing loops

The New Way

- Gate-based structural enforcement
- Blocks tools until context is verified loaded
- Parses stdout with validators
- Bounded logic loops (Max retries)

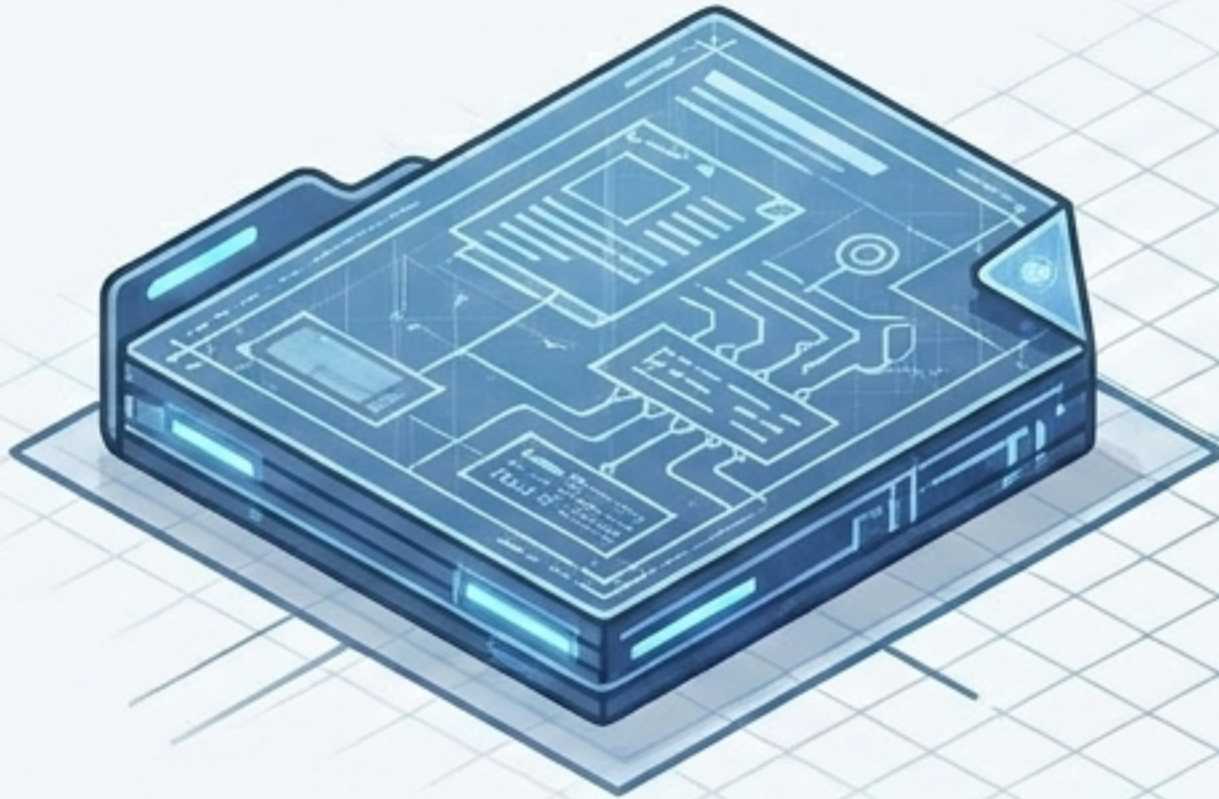
If it's not structurally enforced, it's optional.

The 4-Hook Architecture Engine



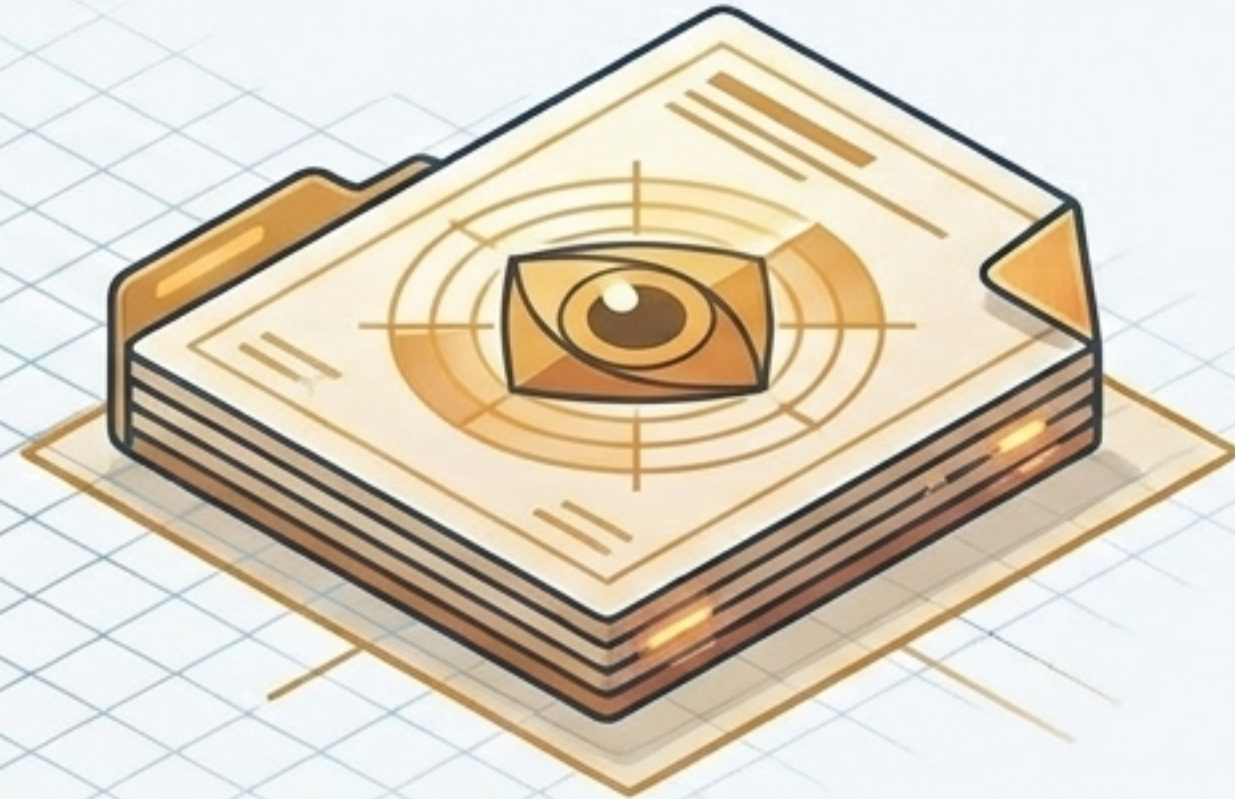
State Management: The Manifest and the Sentinel

`manifest.json`



Generated fresh every session.
Lists all Tier 1/Tier 2 file paths,
sizes, and trigger keywords.

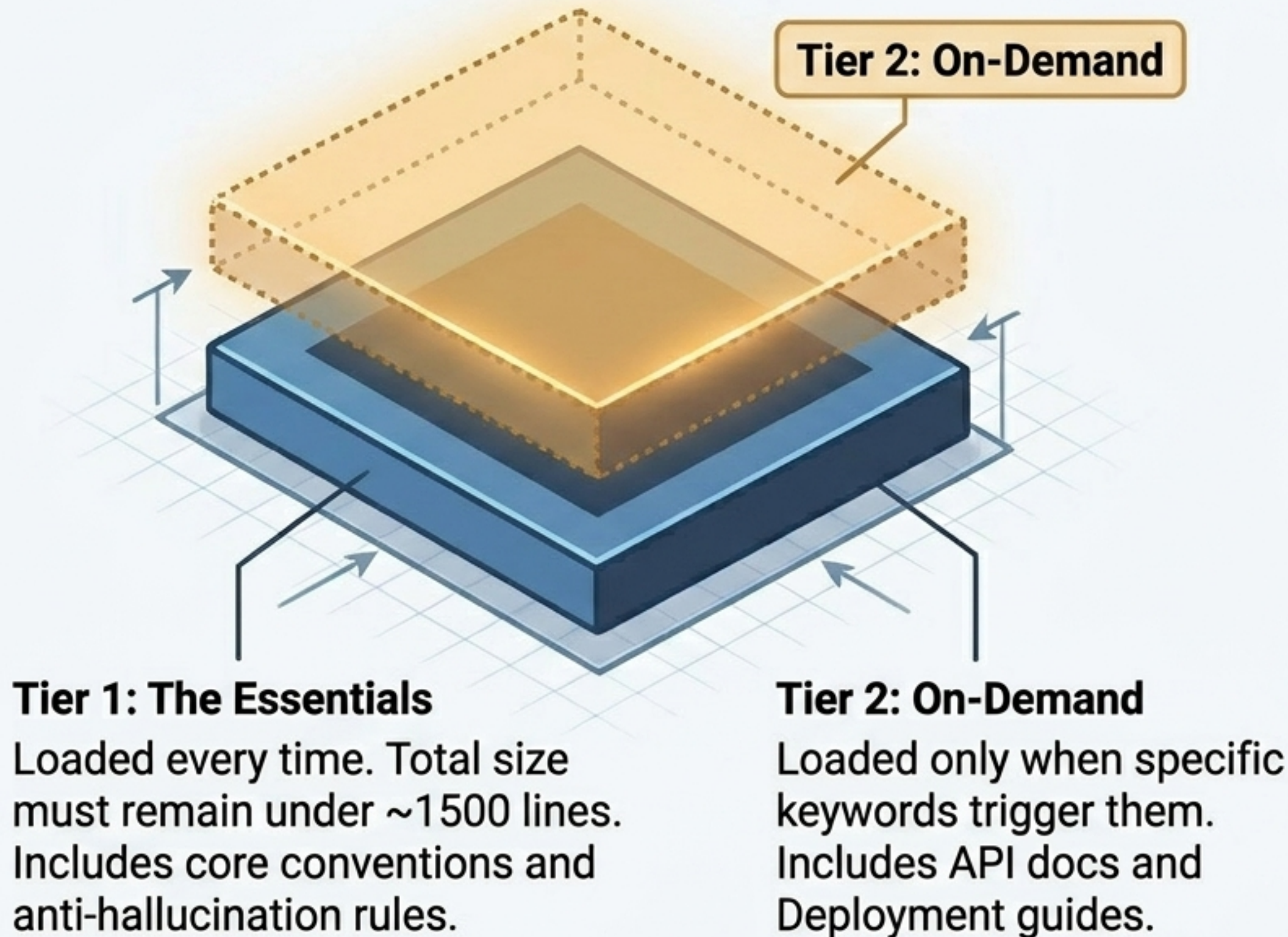
`startup-complete-{session}.json`



The Sentinel.
Tracks actual read actions, Tier
1 status, and cross-checks.

Always suffix with **{session}**. This isolates state, preventing concurrent instances from overwriting each other or resumed sessions reading stale data.

Tiering Strategy: Frequency > Importance

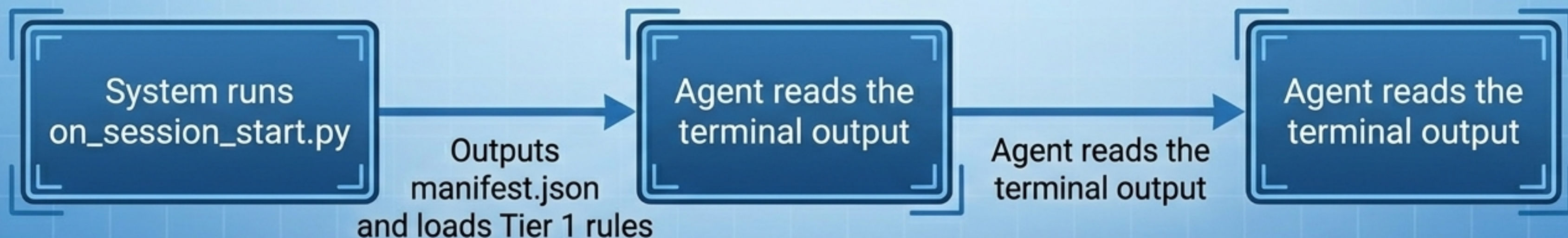


Critical API rules only matter when doing API work.

Put them in Tier 2 to prevent token waste on unrelated tasks.

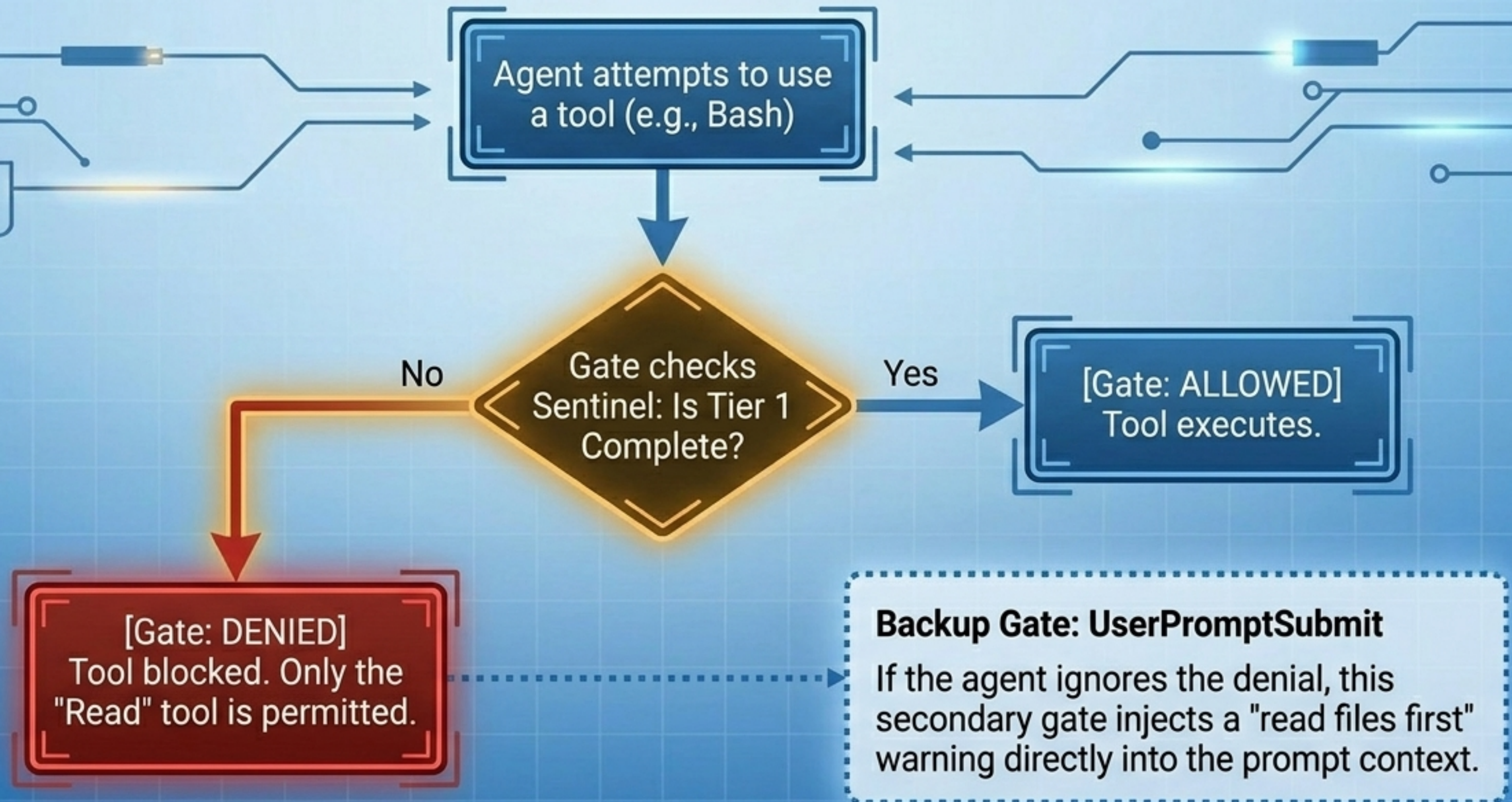
Level 1: The Quick Start

The simplest useful version. It tells the agent what to read, but does not force compliance if the agent decides to skip it.



[Status: No Gates Active. Relies on Voluntary Compliance.]

Level 2: Adding Structural Gates



Level 3: On-Demand Triggers and Drift Detection

Smart Loading

```
Bash('curl api.example.com...')
```

System detects
keyword: 'api'

[DENIED] Tier 2 files
triggered — read
api-rules.md before
proceeding.

Self-Auditing

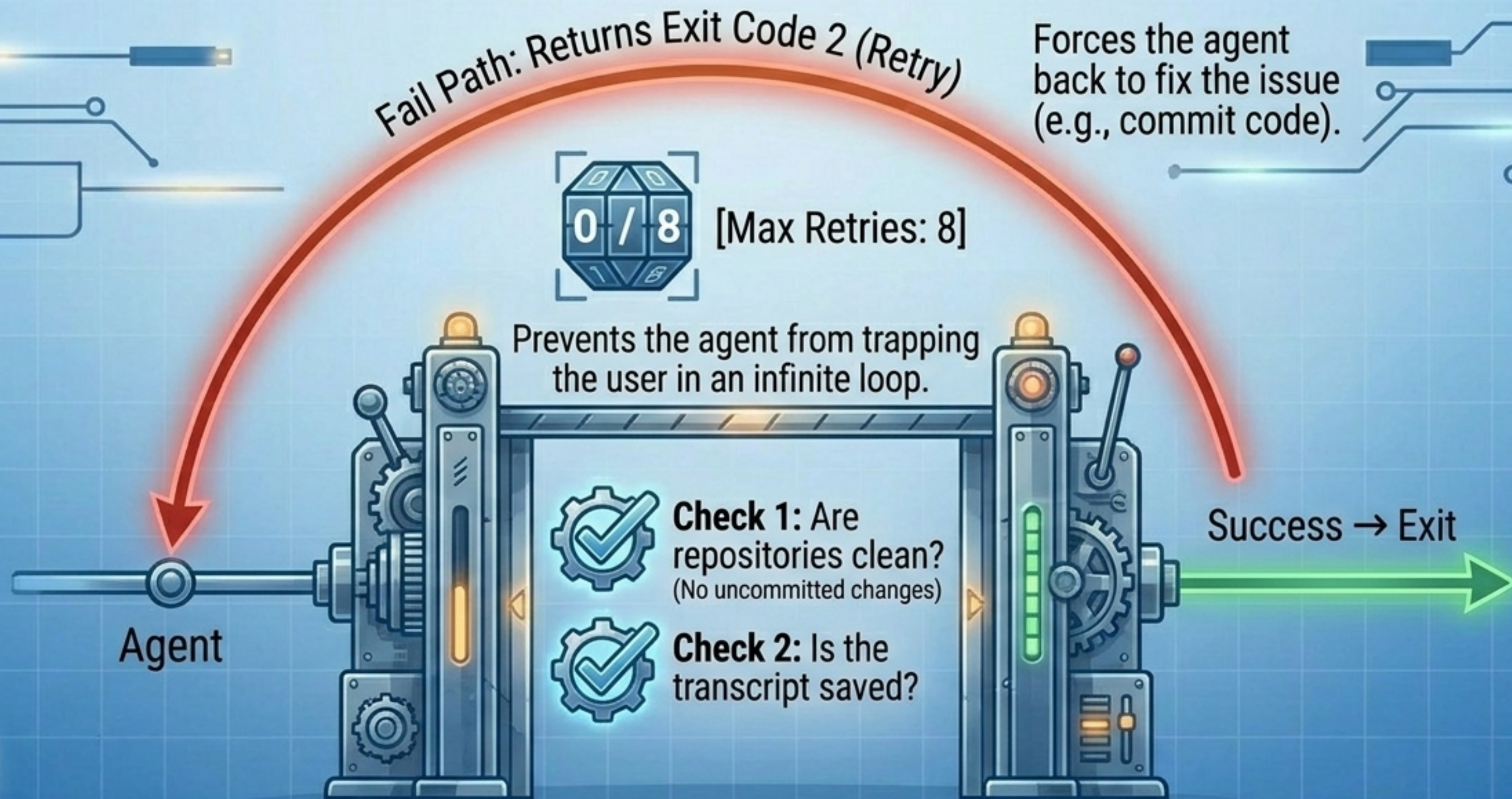
Expected:
58 rules
(Manifest)

Actual:
62 rules
(Disk)



Cross-check runs once per session.
Bounded loops prevent infinite self-healing
spirals. Fix what is safe, log the rest.

Level 4: The Stop Hook Checkpoint



The Lie of the Exit Code

Command Input:

```
git status | grep -v node_modules
```

Terminal Output:

```
M    src/main.py  
??  docs/new-rules.md
```

Exit Code (\$?): 0 (Success)

The exit code only reports that 'grep' ran successfully, masking the actual uncommitted files.

Output-Based Validators

Parse stdout, don't trust exit codes.

empty_output	Passes only if stdout is whitespace.
contains:text	Passes if stdout contains specific string.

Cognitive Defense and Anti-Hallucination



LLMs suffer from a **U-shaped** attention bias—they skip rules buried in the middle of flat lists.

Organize instructions into these 5 distinct phases within Tier 1.

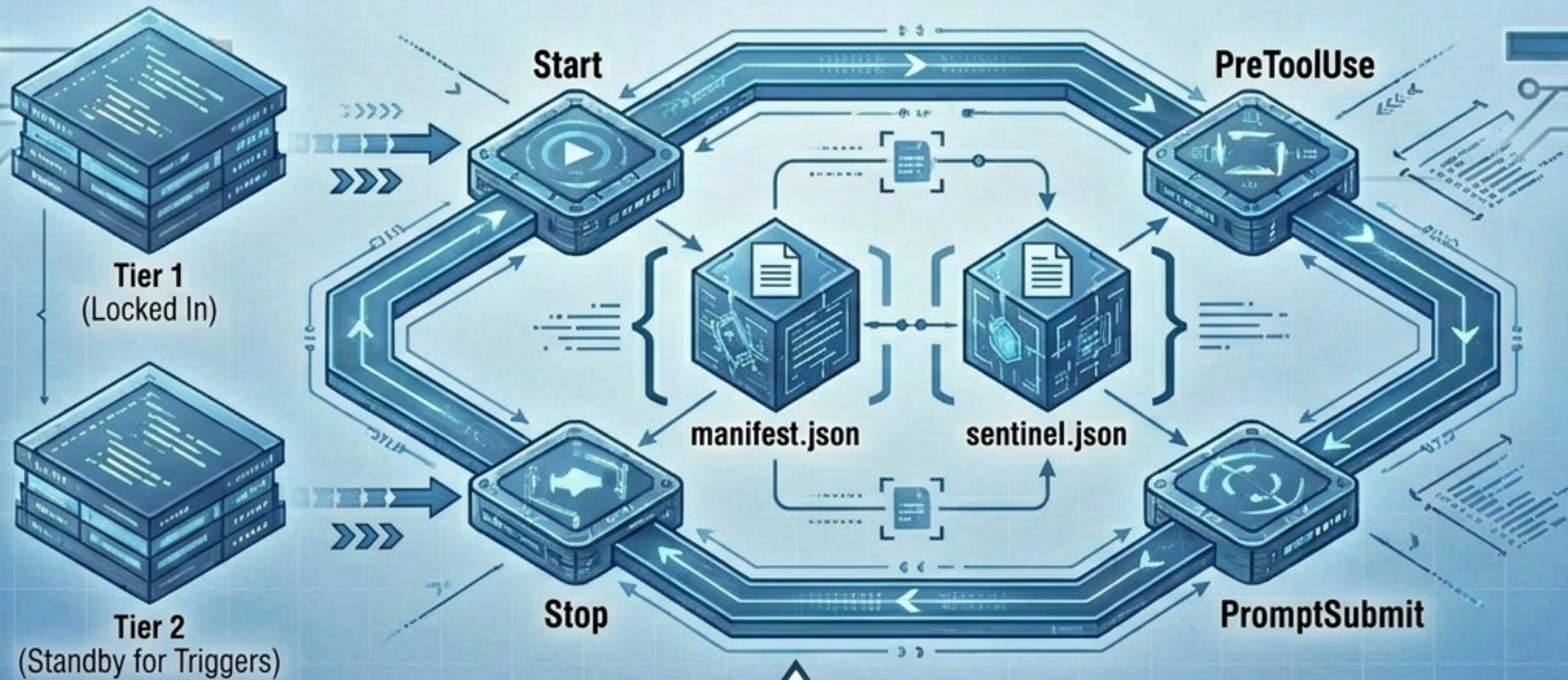
A Framework Agnostic to Platforms

Claude Code	Native Hooks (PreToolUse, SessionStart).
Cursor	.cursor/rules/ + Custom commands for gating.
Windsurf	.windsurfrules + Cascade memories for state tracking.
Aider / Continue.dev	Middleware in the agent loop / Context providers.

Core Universal Principle:

1. Load context ➡ 2. Verify before tools ➡ 3. Sync state ➡ 4. Verify cleanup

The Capstone Architecture



Managed startups yield lower costs, zero hallucinations, and enforced consistency across all AI sessions.

Stop Suggesting. Start Enforcing.



Step 1: Run the Bootstrapping Guide to create your first 5 rules in 15 minutes.



Step 2: Execute the Smoke Test (`python3 tests/smoke_test.py`) to verify all 18 end-to-end hook checks.



Step 3: Complete the 8-Module Mini Course to customize the architecture for your specific repository.

Transform your AI from a chaotic assistant into a highly constrained engineering asset.