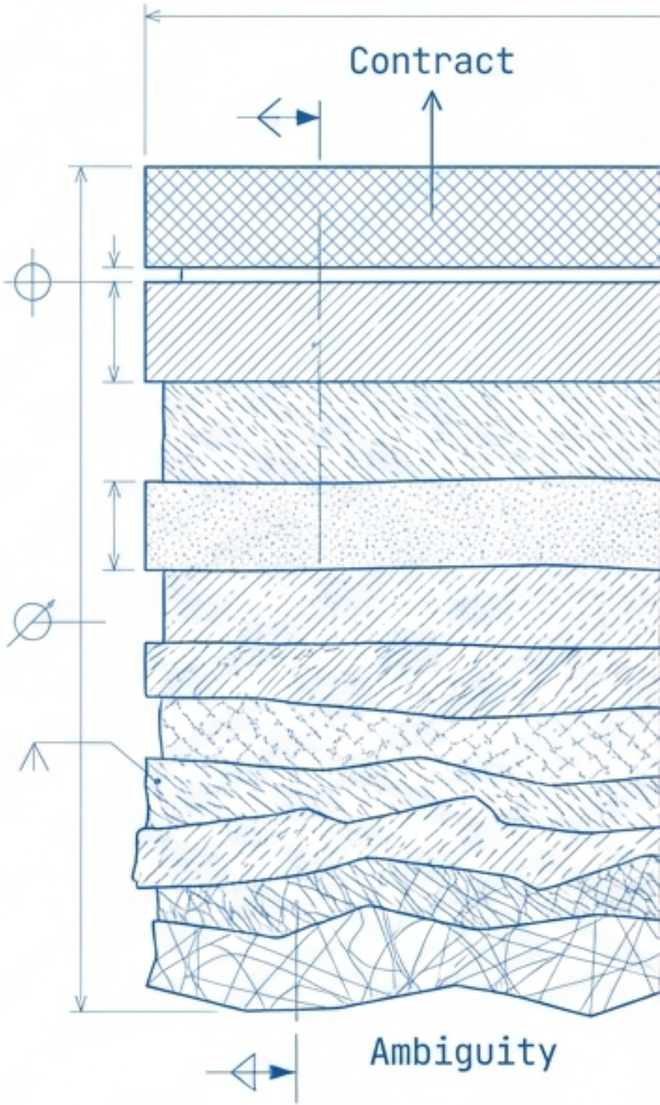
**AUDIT STATUS:** ACTIVE
**TARGET:** JAZZY

# ROS 2 Semantic Truth Audit (Jazzy)

## What’s Specified, What’s De-Facto, What’s Missing — and how rosrustext defines a contract layer.




**CONTEXT:** Spec-First Technical Audit of the ROS 2 Core Framework
**BASELINE:** ROS 2 Jazzy / rclcpp / Nav2
**OBJECTIVE:** Semantic Provenance and Invariant Analysis

Citations:
*Audit Report:* Semantic Provenance and Invariant Analysis for the ROS 2 Core Framework

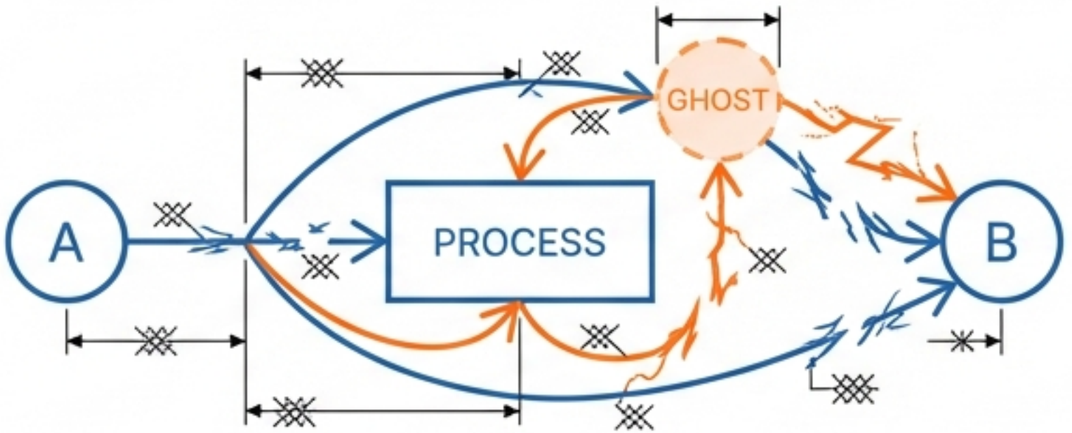
# Why This Audit Exists: The Divergence Risk

## The Promise (DDS/Spec)

ROS 2 was designed for deterministic, safety-critical systems using DDS standards.


## The Reality (De-Facto)

A significant divergence exists between formal design documents (REPs) and operational truth.




1. **Upstream Ambiguity:** Specs often allow multiple execution paths (e.g., 'undefined' terminal states).
2. **Hidden Invariants:** Critical behaviors (thread safety, atomicity) are buried in `rclcpp` source code, not specs.
3. **The Consequence:** 'Vibe Coding' vs. **Engineering**. Alternative client libraries drift from the 'Professional Stack' behavior, leading to non-determinism and silent failures.

*Audit Report:* Introduction | *Methodology:* Context: Why methodology matters [methodology.md] | *Methodology:* Parity is judged by observable ROS behavior, not API similarity [methodolo

# The Baseline: Jazzy + rclcpp + Nav2


**Why this specific combination?** It represents the “Professional Stack”—the de-facto standard for mobile robotics production.

## The Components

**rclcpp** defines **“correct” behavior**, often exceeding the **REPs**. **Nav2** is the primary consumer that enforces this behavior.

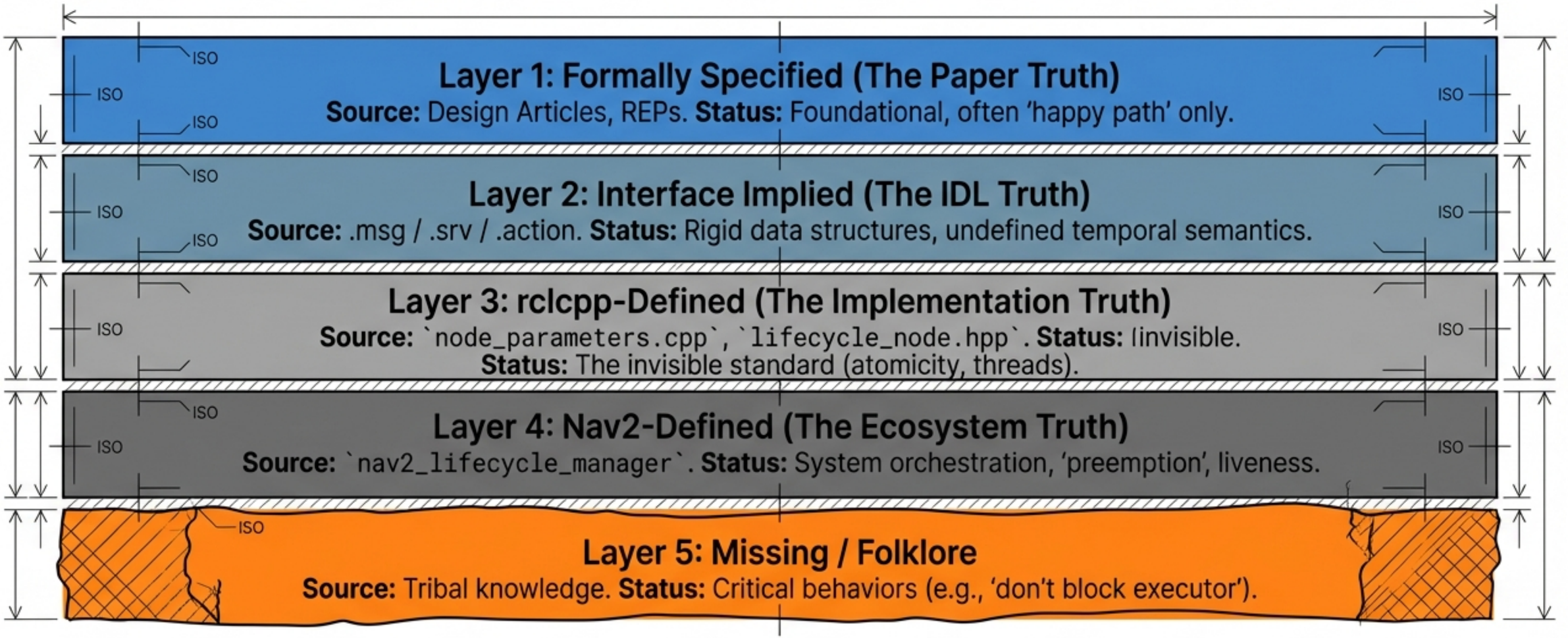
## The Integration Reality

Nav2 assumes `rclcpp` **quirks are standard features**. A node that complies with specs but fails Nav2 expectations is effectively broken.

**Audit Goal**: Elevate “folklore“ and implementation details to normative requirements.

*Audit Report*: The Professional Stack (Jazzy + Nav2) introduces invariants...
*Professional Stack Spec*: rclrs + rosrustext_rclrs should be a drop-in professional alternative to rclcpp [professional_stack.md]
*Nav2 Documentation*: https://docs.nav2.org/

# Taxonomy of 'Truth' in ROS 2

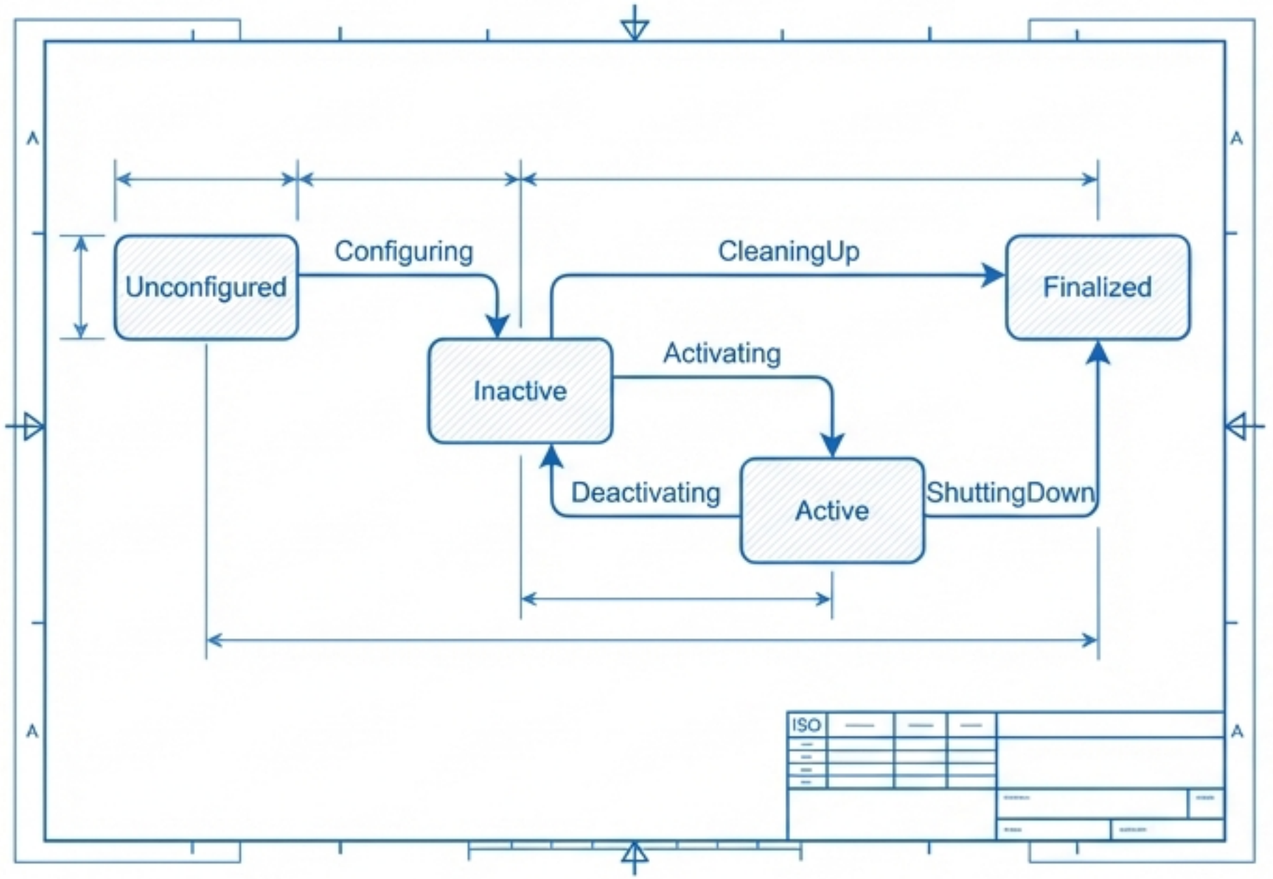



*Audit Report:* The Managed Node Lifecycle: Formal and De-Facto Provenance
*Audit Report:* Missing-Spec Inventory: Critical Gaps and Folklore

Forensic Engineering

# Lifecycle — What's Formally Specified

## The "Happy Path" defined in Design Articles




**Interface Constraints:**

- Lifecycle services (~/get_state, ~/change_state) are the normative control mechanism.
- Statelessness: `Unconfigured' must have no active comms (clean slate).
- Communication Gating: Data flow (pubs/subs) is only permitted in `Active`.

**Source Provenance:**

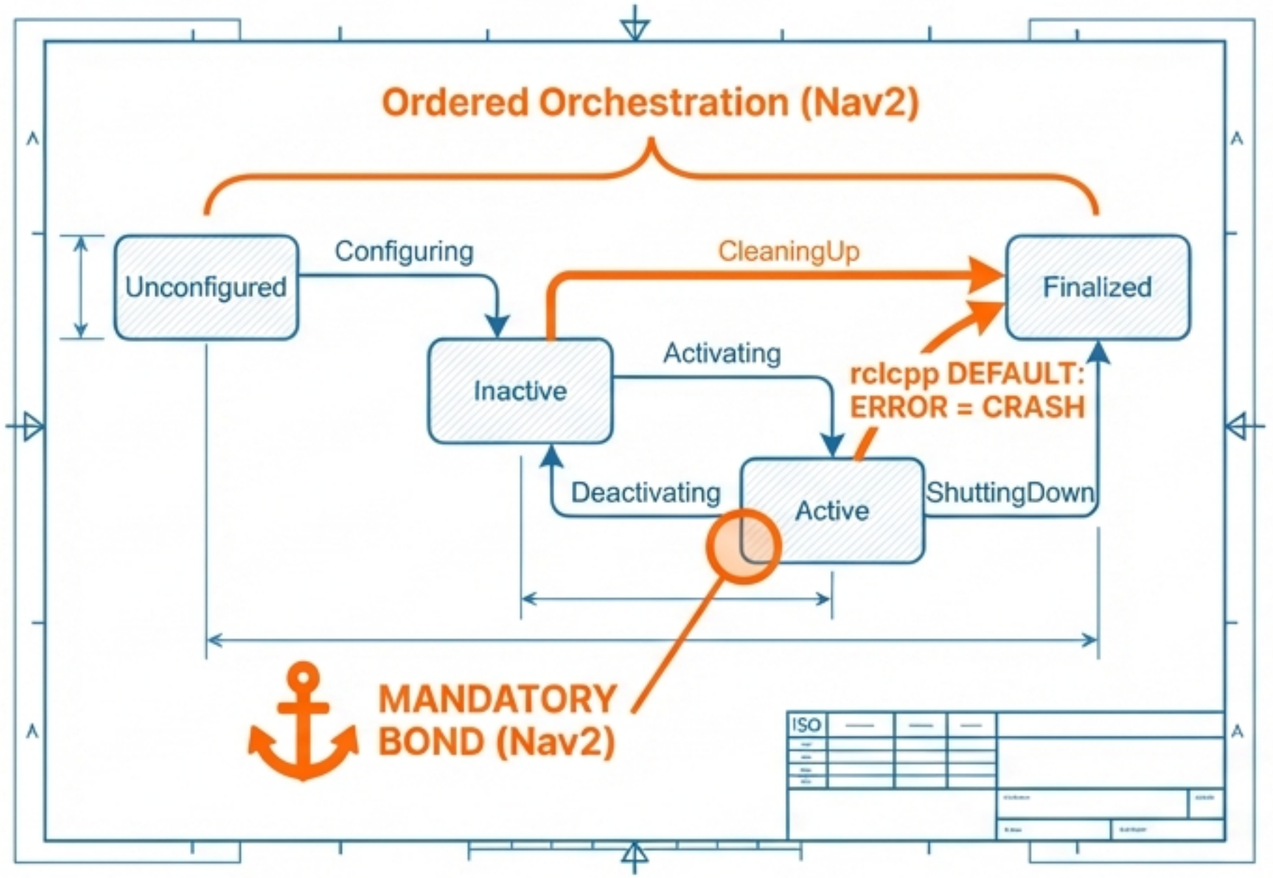
- These rules are stable and explicitly defined in ROS 2 Design Articles.

*Audit Report*: Rule-Level Provenance for lifecycle_core.md
*ROS 2 Design*: Managed Nodes https://design.ros2.org/articles/node_lifecycle.html
*Lifecycle Spec*: Required Semantics (Normative) [lifecycle.md]

Forensic Engineering

# Lifecycle — De-Facto (rclcpp/Nav2) & Missing

Where the ‘Professional Stack’ diverges from the design.




**De-Facto Rule: Mandatory Bond (Nav2)**

- Rule: Active nodes *must* maintain a Bond heartbeat.
- Risk: Without this, ‘silent failure’ crashes the robot stack.

**De-Facto Rule: Deterministic Orchestration (Nav2)**

- Rule: Bringup is ordered (Sensors -> TF -> Controller).
- Provenance: Nav2 System Tests.

**Missing Upstream: Error Routing (rclcpp)**

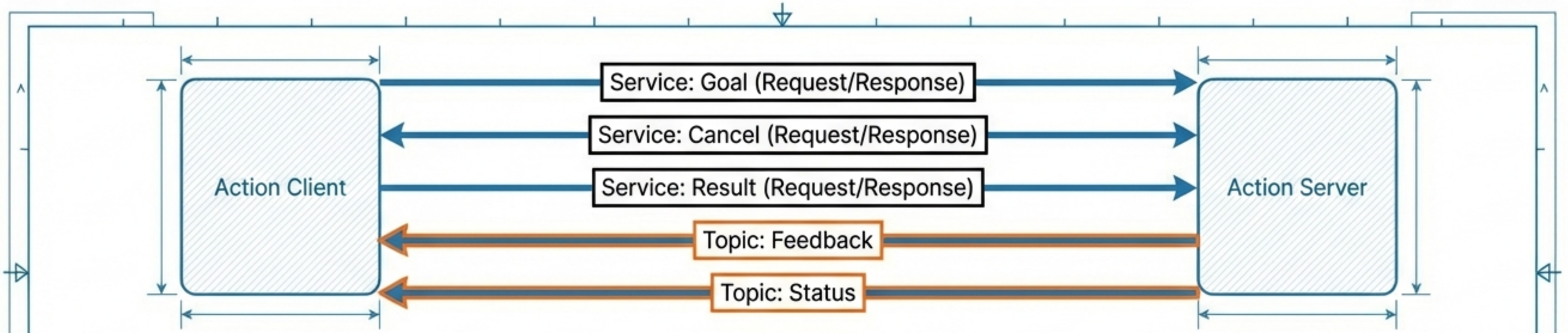
- Folklore: `on_error` callback defaults to `Finalized`.
- Implication: Custom implementations failing this route create ‘zombie nodes’.

*Audit Report:* Bond-based heartbeats are required… Defined by Nav2
*Nav2 Docs:* Lifecycle Manager https://docs.nav2.org/configuration/packages/configuring-lifecycle.html
*rclcpp Source:* Default onError behavior https://docs.ros.org/en/jazzy/p/lifecycle/

Forensic Engineering

# Actions — What's Formally Specified

## The Mechanics, Identity & State, and Visibility defined in Design Articles.




**The Mechanics:**

- Three Services: Goal, Cancel, Result.
- Two Topics: Feedback, Status.

**Visibility:**

- Action Servers must cache results for Status topic is the 'source of truth'.

**Identity & State:**

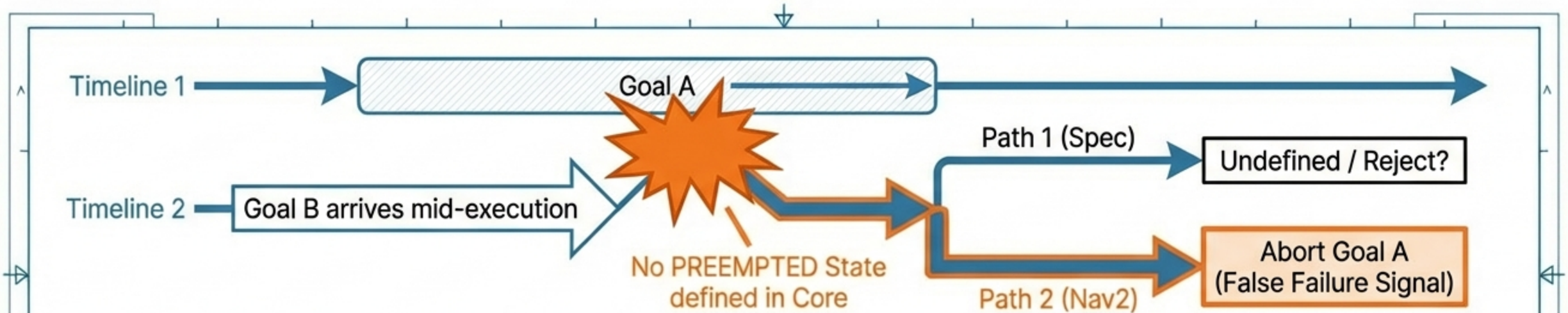
- Goal IDs must be UUIDs (generated by Client).
- Terminal States: `SUCCEEDED', `ABORTED', `CANCELED'.

for late-joining clients (s (Transient Local QoS).
of truth' for introspection.

*Audit Report*: Rule-Level Provenance for action_core.md
*ROS 2 Design*: Actions https://design.ros2.org/articles/actions.html
*Action Spec*: Required Semantics (Normative) [action.md]

Forensic Engineering

# Actions — The Semantic Gap (Preemption)




**The Conflict:**

- Nav2 requires **Preemption** (New Goal overrides Old Goal).
- ROS 2 Action State Machine has **no PREEMPTED state.**

**The 'Shadow' Terminal State:**

- Observation: Nav2 servers default to `ABORTED` upon replacement.
- Risk: `ABORTED` signals system failure, but preemption is a strategic choice. This ambiguity confuses logic.

**Implementation Invariants (Threading):**

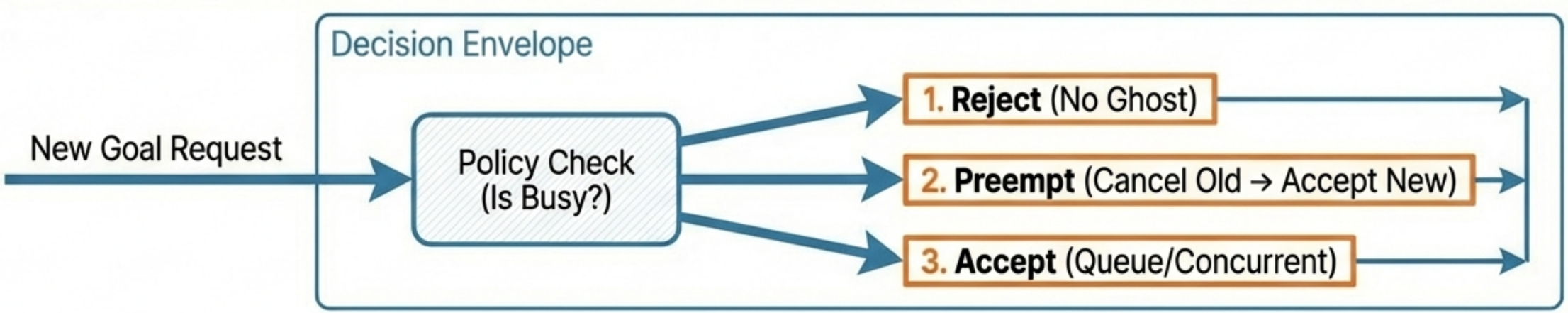
- Folklore: Execute callback *must* be threaded.
- Risk: Synchronous blocking stalls the DDS executor and starves heartbeats.

*Audit Report:* The Specification Gap: Preemption vs. Abortion
*Nav2 Issue:* Add support for preemption in actions https://github.com/ros2/design/issues/284
*Audit Report:* Action Threading… defined by rclcpp behavior

Forensic Engineering

# Actions — How action_core.md Defines the Contract

The Spec-First Fix: `action_core.md` formalizes missing logic.




## Decision Envelopes:

- Outputs are `SendGoalDecision` (Outcome), NOT just transport errors.
- **No Ghosts**: A rejected goal *must not* emit status/feedback later.

## The Preemption Invariant:

- Forces a deterministic policy choice (reject new vs. cancel old) rather than undefined behavior.

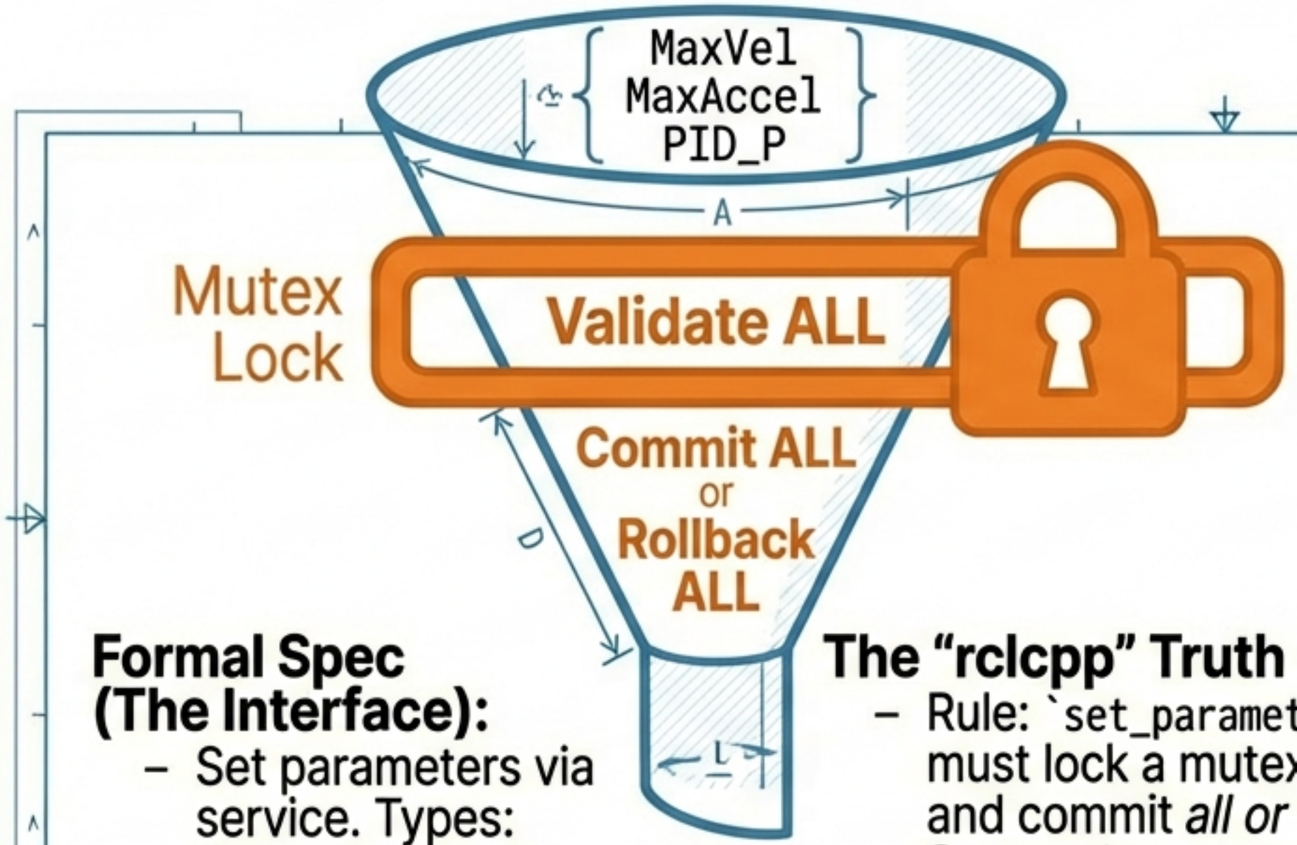
## Invariants Enforced:

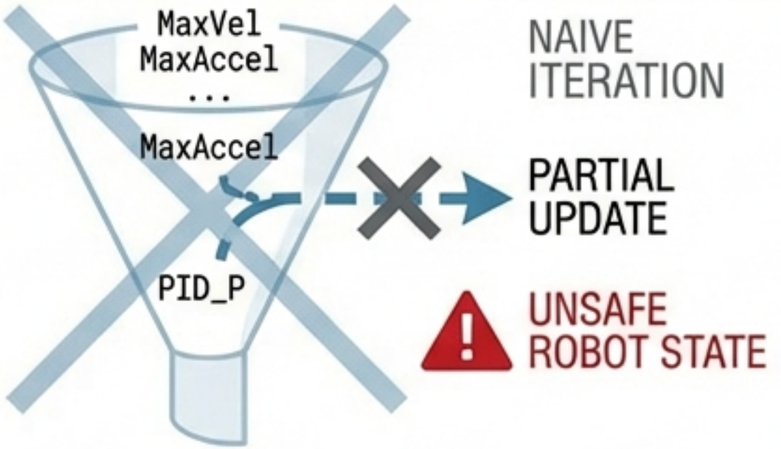
- Status Sequence Monotonicity (Time cannot go backward).
- Terminal Immutability (Once `ABORTED`, cannot change to `SUCCEEDED`).

*Action Core Spec:* Decision envelopes…. Policy outcomes are not errors [action_core.md]
*Action Core Spec:* Cancel race validation [action_core.md]
*Audit Report:* Explicit Preemption Policies

Forensic Engineering

# Parameters — Formally Specified vs. rclcpp-Defined




**Formal Spec (The Interface):**

- Set parameters via service. Types: Bool, Int, Double, String, Arrays.

**The "rclcpp" Truth (Atomicity):**

- Rule: `set_parameters_atomically` must lock a mutex, validate *all*, and commit *all or nothing*.
- Source: `node_parameters.cpp` implementation artifact.
- Risk: Naive iteration allows partial updates (e.g., Velocity updates, Acceleration fails = Unsafe Robot).

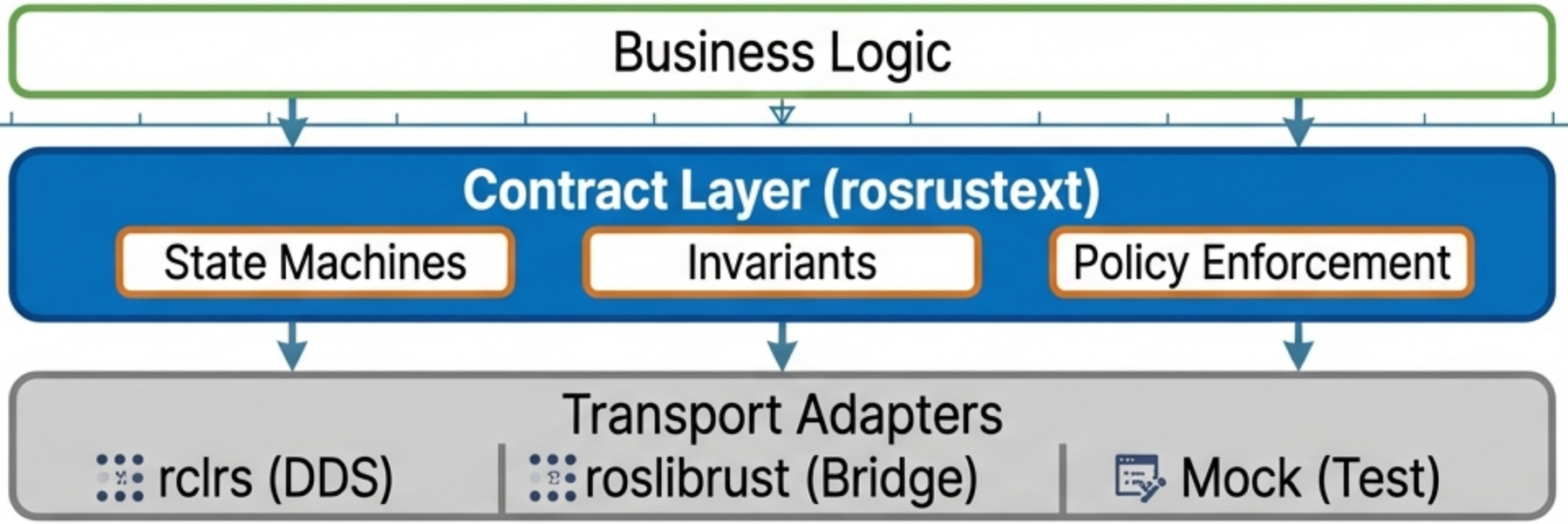
**The Callback Invariant:**

- Rule: Parameter callbacks run in the node's "Mutual Exclusion Group".
- Risk: Updating parameters blocks timers/subscriptions (Deadlock potential).

*Audit Report:* The 'rclcpp' Definition of Atomicity
*rclcpp Source:* `node_parameters.cpp` https://github.com/ros2/rclcpp/blob/rolling/rclcpp/src/rclcpp/node_interfaces/node_parameters.cpp
*Parameter Core Spec:* Atomic sets... apply all changes, or apply none [parameter_core.md]

Forensic Engineering

# The Solution — rosrustext Contract Layer




**Methodology: Spec-First, Transport-Agnostic.**

- We do not port code; we port *semantics*.
- **`rosrustext_core`**: Pure Rust state machines. Zero ROS dependencies.

**Codifying the Folklore:**

- What `rclcpp` does implicitly (mutex), `parameter_core` enforces explicitly.
- What `Nav2` expects (Bond), `lifecycle_core` validates via invariants.

**The Firewall:**

- Separates "Logic" from "Plumbing". Switching transports does not break system behavior.

*Methodology:* Parity extension, not a rewrite [methodology.md]
*Methodology:* One semantic truth… No adapter is allowed to invent its own state machine [methodology.md]
*Audit Report:* The necessity of stricter contracts

# What Becomes Possible?

- **Verifiable Safety Arguments:** Prove a node *cannot* publish while Inactive (compiler-enforced).
- **Drift Detection:** Contract tests catch divergence between `rclcpp` (JetBrains Mono) and `rosrustext` (JetBrains Mono) before deployment.
- **Adapter Conformance:** Swap underlying transport (DDS vendor, Zenoh, rosbridge) without altering semantic behavior.

## Conclusion: Documentation is not enough. You need a Contract Layer to survive the "Professional Stack".

**Audit Report:** Conclusions and Recommendations
**Professional Stack Spec:** Definition of Done... Rust nodes are operationally indistinguishable [professional_stack.md]
**Executor Spec:** Definition of Done... No work happens without explicit spinning [executor.md]

Forensic Engineering

# Appendix — Traceability Snapshot

Representative rules derived from the Audit and mapped to the Core Contract.

| Rule | Classification | Primary Upstream Citation | Notes / Risk |
|---|---|---|---|
| Active Node Liveness | Nav2-Defined | [Nav2 Config URL] | Requires `Bond`; missing upstream. **Risk:** Silent failure. |
| Action Preemption | Ecosystem (Folklore) | [Nav2 Issue #284 URL] | No `PREEMPTED` state. Ambiguous termination. |
| Param Atomicity | rclcpp-Defined | [rclcpp node_parameters.cpp URL] | Batch updates must use mutex. **Risk:** Partial state. |
| Lifecycle Transitions | Formal Spec | [Design: Managed Nodes URL] | State machine structure defined in design articles. |
| Statelessness | Formal Spec | [Design: Managed Nodes URL] | Unconfigured must have no pubs/subs. **Risk:** Ghost data. |
| Execution Threading | rclcpp-Defined | [rclcpp Issue #2912 URL] | Callbacks run in Mutex groups. **Risk:** Deadlocks. |
| Ghost Goals | Missing Spec | [action_core.md] | Rejected goals must not emit feedback. |
| Transition Error | rclcpp-Defined | [rclcpp Lifecycle Docs URL] | `on_error` defaults to `Finalized`. **Risk:** Inconsistent recovery. |

*Audit Report:* Rule-Level Provenance Tables.