SINGULARICS

White Paper

Intent: The Organizational Intent Control Plane

How organizational intent, human judgment, and agent orchestration operate through one governed control plane.

Alexander S. Petty
SINGULARICS | singularics.ai

First published August 2026.

singularics.ai

Contents

1 The Control Plane 4

2 The Four Things 6
- 2.1 The Graph: the nouns . . . 6
  - 2.1.1 Levels, vocabulary, and why “Program” is not a node kind . . . 7
- 2.2 The Command Spine: the verbs . . . 8
- 2.3 Projections: the views . . . 8
- 2.4 Judgment Gates: the governance . . . 9

3 The Work Lifecycle 10

4 How the Graph Maps to the Operating Model 11

5 How Judgment Gates Enforce the Model 12
- 5.1 No execution without permission . . . 12
- 5.2 No acceptance without evidence . . . 12
- 5.3 Bounded authority is enforced by the envelope . . . 12

6 The Command Spine as the Evidence Trail 13

7 How Agent Execution Works 14
- 7.1 The native runtime . . . 14
- 7.2 External agents via MCP . . . 14

8 The Gap the Platform Fills 15
- 8.1 The graph must be explicit . . . 15
- 8.2 The gates must be enforced . . . 15
- 8.3 The evidence trail must be automatic . . . 16
- 8.4 Agent coordination must be mechanical . . . 16

9 How the Platform Connects to the Operating Model 16

10 The Tracker, the Spreadsheet, and the Ungoverned Orchestrator 17

11 Three Mechanisms in the Control Plane 18

Abstract

An operating model is a set of decisions about who decides, how fast, and on what evidence. Running that model at scale requires software, because humans cannot hold the graph of intent, enforce the judgment gates, maintain the evidence trail, and coordinate agent execution across a portfolio in their heads. Intent is the control plane that lets AI execution capacity serve declared human purpose, under human judgment, with everything attributed. This paper defines Intent's four structural elements (the Graph, the Command Spine, Projections, and Judgment Gates), the governed work lifecycle with its seven states, the MCP protocol that lets any agent executor connect under the same governance rules, and the design principle that keeps the graph core stable as enterprise domains land on top of it. It shows how each element maps to the operating model and explains why the platform exists: not as a tool that assists the model, but as the mechanism that makes it runnable.

The Control Plane

The AI-Native Operating Model can be run with spreadsheets and discipline for a single team. It cannot be run that way for a portfolio. We learned this directly: the first organization that adopted the model across five simultaneous efforts spent more time coordinating governance artifacts than it saved from AI-driven delivery. The governance trail was manual, the intent graph was implicit, and the judgment gates were cultural. Under pressure, every one of them failed. Intent exists because that experience proved the model needs a mechanism, not just a method.

Intent is a control plane that lets AI execution capacity serve declared human purpose, under human judgment, with everything attributed.

That sentence is precise and every word carries weight.

Control plane. Intent does not do the work. It governs what work is done, by whom, under what authority, and whether the result is accepted. The work itself is performed by agents operating under the platform's direction.

Declared human purpose. Nothing exists in the system that was not declared by a human as worth pursuing. Work exists only in service of declared intent. An output with no linked outcome is rendered, flagged, and treated as a defect of alignment, not as a category of work.

Under human judgment. No execution starts without human permission. No result is accepted without human judgment or a standing, human-authored, audited, revocable policy. The platform enforces this at the gate level, not as a best practice.

Everything attributed. Every write carries an actor. Every state transition carries its source. Every acceptance carries who decided and what evidence they reviewed. The platform does not have a mode where things happen without a trail.

> *Agents bring capacity. The graph brings purpose. The platform only lets the first serve the second.*

The Four Things

Every piece of the system is one of four things. Knowing which one you are touching tells you the rules that apply.

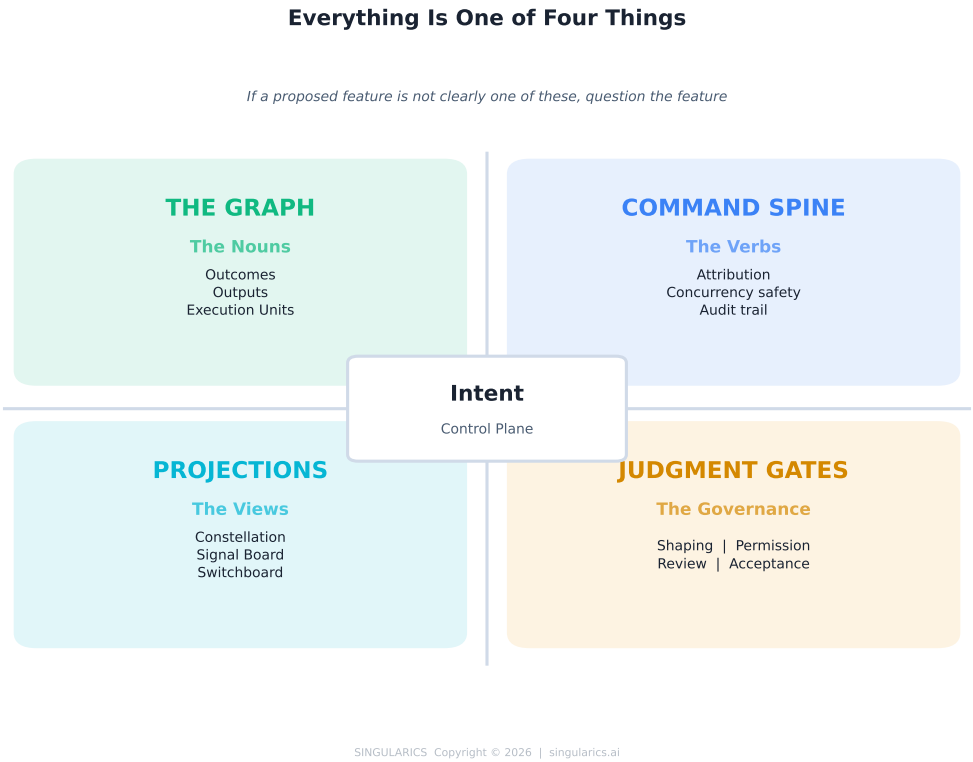

Figure 1: The four structural elements of Intent. The Graph holds the nouns. The Command Spine carries the verbs. Projections render the views. Judgment Gates enforce governance. If a proposed feature is not clearly one of these, question the feature before building it.

The Graph: the nouns

The graph is the product. Everything else is in service of it. It holds what the organization intends and what exists to serve that intent.

Three node types carry the structure.

Outcomes are intended state changes. They are the only reason work exists. In the operating model, outcomes correspond to strategic intent: the things the Intent Council has placed on the Now, Next, or Later horizons.

Outputs are deliverable artifacts that serve outcomes. In the operating model, outputs are the units of work the Flow Council allocates capacity against and Decision Owners accept daily.

Execution units are granular, agent-assigned work under an output. They are the tasks the Fleet Lead decomposes from the daily spec and assigns to the agent fleet.

Edges connect them: contributes, depends on, blocks, and association flows. The edge vocabulary is closed and small on purpose, because a graph with too many edge types becomes unreadable and ungovernable.

> *An output with no linked outcome is an orphan. The platform renders it, flags it, and treats it as a defect of alignment. Work that serves no declared intent is not a category. It is a mistake made visible.*

Levels, vocabulary, and why "Program" is not a node kind

Strategic, operating, and artifact labels come from admin vocabulary and level configuration, never hard-coded. This is a design principle, not merely a convenience.

A "Program" is not a node kind. It is an outcome at a level the Office governs. This distinction is what let the Portfolio Office land as 8,400 lines of code with zero changes to the graph core, and it is the template for absorbing the next enterprise domain. When a new organizational concept needs to be represented, it maps onto the existing graph as a level, a record, or a vocabulary entry. It does not become a new parallel structure.

The implication is that the graph core is stable. New domains arrive as configuration rather than as schema changes. The governance rules, the command spine, the judgment gates, and the lifecycle all apply unchanged to the new domain, because the new domain is expressed in the same structural vocabulary the platform already understands.

8,400 lines of code to land the Portfolio Office domain on Intent, with zero changes to the graph core. That ratio, a full enterprise domain for under 10,000 lines and no structural migration, is the proof that the abstraction holds.

This is how the platform scales from a single team to a portfolio of hundreds of agents across multiple governance layers without the graph core growing in complexity.

The Command Spine: the verbs

There is one way to change the graph: a mutation command executed through the command spine. This is the single path by which anything changes, and it carries four guarantees.

Attribution. Every command carries an actor. No anonymous writes. The operating model requires that every acceptance, every state change, and every escalation be traceable to a named person. The spine is how that requirement is mechanized.

Concurrency safety. Version-pinned commands use compare-and-set on the node's version at the update itself. Two writers cannot silently overwrite each other. The loser receives a stale-command error, not a silent merge. This has been proven under concurrent agent fleets and holds at high call rates without degradation.

Audit. Writes emit to the audit log, the assertion log, and the governance ledger in the same transaction. The operating model demands that status be read, not written. The spine is what makes the ledger the status.

Sourced transitions. State transitions carry their source: user, runtime, runtime failure, integration, reactor, or MCP agent. The operating model distinguishes between human decisions and machine actions, and the spine encodes that distinction at the data level.

If code writes to graph tables without going through the spine, that is a bug regardless of whether it produces the correct result.

Projections: the views

Every screen, every dashboard, and every read API is a projection of the graph plus runtime state. Projections never carry truth of their own. If a projection disagrees with the graph, the projection is wrong.

This principle is what lets the platform serve multiple audiences from a single source of truth. The Intent Council sees the constellation view, where outcomes are rendered as a structural map of strategic intent. The Flow Council sees the signal board, where the lifecycle is rendered as lanes and the human judgment queue sits at the top. The Fleet Lead sees the operations switchboard, where agent telemetry, execution packets, and claims are visible in real time.

Each projection is a different lens on the same underlying graph. None of them adds information that the graph does not contain. None of them can change the graph directly. Changes go through the spine.

Judgment Gates: the governance

Judgment gates are the points where work cannot proceed without a human decision or a standing, human-authored policy.

Four gates govern the work lifecycle.

The shaping gate is structural. Work cannot leave the backlog without linked intent, a description, and acceptance criteria. The system states what is missing in plain language. This gate corresponds to the spec step in the operating model: the Decision Owner cannot sign a spec that the platform considers incomplete.

The permission gate is human. A named person approves work for execution. No agent, native or external, can execute work a human has not approved. This is the concrete mechanism behind the operating model's rule that humans own whether and what, while agents own how.

The review gate requires evidence. Executors submit work with reports and evidence. Review cannot be entered without a work report. This is the harness step of the operating model: the Decision Owner judges against evidence, not against a demo.

The acceptance gate is human. Verify and close. Acceptance is information: it updates outcome conditions, which updates alignment, which changes what comes next. This is the judge step of the operating model, and it is the step that closes the daily cycle.

The standing rule across all four gates: humans own whether and what. Agents own how. Nothing crosses from declared to executing, or from review to accepted, without a live human judgment or a standing, human-authored, audited, revocable policy.

The strongest objection to this architecture is that the gates slow agents down. They do not. The gates add latency of seconds, not hours, because they fire at state transitions that already require a decision. The Decision Owner who judges the increment does not wait for the platform to finish. The platform waits for the Decision Owner to decide. The bottleneck is human judgment, which is the constraint the entire model is designed around, and the gates simply make that constraint visible and enforceable rather than invisible and unmanaged.

The Work Lifecycle

One governed lifecycle, shared by humans, the native runtime, and external agents:

> **Declared → Shaping → Ready → Executing → Review → Verified → Closed**

Needs Attention is the alarm rail, reachable from any active state when the work requires human intervention that the normal flow does not provide.

Each transition is governed by a gate, and the gates map directly to the operating model's daily cycle.

Declared to Shaping. Work enters the system as declared intent. The shaping gate requires linked intent, a description, and acceptance criteria before work can proceed. This corresponds to the spec step: the Decision Owner cannot sign a spec that the platform considers incomplete.

Shaping to Ready. The shaping gate is satisfied. Criteria are checkable, constraints are stated, and scope is bounded. The spec is signed. This corresponds to the signed spec that starts a build day.

Ready to Executing. The permission gate fires. A named person approves work for execution. No agent can execute work a human has not approved. This is the concrete mechanism behind the rule that humans own whether and what, while agents own how.

Executing to Review. The executor submits a work report with evidence. The review gate cannot be entered without that report. This corresponds to the harness step: the Decision Owner judges against evidence, not against a demo.

Review to Verified. The acceptance gate fires. A human reviews the evidence and accepts. Acceptance updates outcome conditions, which updates alignment, which changes what comes next. This is the judge step that closes the daily cycle.

Verified to Closed. The outcome is met and the work is closed. The Flow Council confirms closure at weekly calibration, and capacity returns to the pool.

The lifecycle is enforced by the state machine. There is no transition that bypasses a gate. There is no mode where work moves from Declared to Executing without passing through Shaping and the permission gate. The platform does not have a bypass mode, because the operating model does not have one.

How the Graph Maps to the Operating Model

The mapping is direct and deliberate.

Graph Element	Operating Model Concept	Governed By
Outcome	Strategic intent, horizons	Intent Council
Output	Unit of work, daily acceptance	Decision Owner
Execution unit	Agent task, fleet work	Fleet Lead
Edges	Dependencies, contributions	Graph policy
Orphaned output	Alignment defect	Platform (flagged)

The three horizons are a projection of outcomes. Now, Next, and Later are states in the outcome lifecycle, not separate lists. The No Owner, No Now gate is enforced by the platform: an outcome cannot enter the Now state without a named Decision Owner linked to it.

The daily cycle is a loop through the output lifecycle. Specify maps to the shaping gate. Build maps to execution. Judge maps to the review and acceptance gates. Land maps to the state transition that the spine records.

The weekly calibration is a projection of Now-state outcomes against capacity. The quarterly rebalance is a projection of all outcomes against strategic intent. Neither requires a separate artifact. Both read the graph.

How Judgment Gates Enforce the Model

The operating model makes several claims that only hold if they are enforced mechanically rather than culturally. The judgment gates are the enforcement mechanism.

No execution without permission

The permission gate prevents any agent from executing work that a human has not approved. This is not a workflow suggestion. It is a database-level constraint. An execution packet cannot be dispatched for an output that has not transitioned through the permission gate. The platform does not have a bypass mode.

No acceptance without evidence

The review gate requires a work report before the output can enter review. The Decision Owner cannot accept an output for which no evidence has been submitted. This mechanizes the harness requirement: acceptance is a judgment made against evidence, not against a walkthrough.

Bounded authority is enforced by the envelope

The Decision Owner's acceptance authority is bounded by the envelope the Intent Council sets at the start of the outcome. The platform records the envelope and surfaces escalation requirements when an acceptance would exceed it. The Decision Owner does not need to remember the boundaries. The platform remembers them.

> *The gates are not guardrails that the organization hopes people will follow. They are constraints the platform enforces. The difference is the difference between a policy document and a locked door.*

The Command Spine as the Evidence Trail

The operating model demands that status be read, not written. Nobody compiles a status deck. The evidence ledger is the status, and every governance forum opens by reading it rather than hearing it summarized.

The command spine is what makes this possible. Every mutation is recorded with its actor, its timestamp, its source, and its effect. The audit log is not a separate system that someone must remember to update. It is a byproduct of the spine's operation.

> **0** status reports required. The command spine produces the evidence trail as a byproduct of normal operation. Every governance forum reads the ledger directly.

In practice, this eliminates an average of four to six hours per week per team that was previously spent compiling, reviewing, and presenting status artifacts. Across a portfolio of ten teams, that is 40 to 60 hours per week returned to judgment and spec work.

For a regulated environment, this is a stronger evidentiary position than a phase-gated process. A phase-gated process produces documents describing intent. The spine produces a continuous record of what actually happened, who decided, and on what evidence. The difference shows when someone asks to reconstruct the decision history for a specific change.

How Agent Execution Works

Two kinds of executor operate under the same governance rules, and neither can bypass the judgment gates.

The native runtime

The platform dispatches execution packets to its agent worker. Packet status, output state, review state, and signal board state must agree. Disagreement between them is treated as a bug, not as a state to be resolved manually.

External agents via MCP

Any MCP-compatible client with an API key can connect to Intent as an external executor. The protocol is a five-step loop, and every step is governed by the same rules that apply to the native runtime.

Available work. The agent queries the market surface to see what work is available for execution. The market is a projection of the graph filtered by the agent's tier and channel. An observer-tier key sees read-only tools. A builder-tier key sees execution tools. The agent cannot see work it is not authorized to claim.

Claim. The agent claims a unit of work. Claims are atomic and compare-and-set: two agents claiming the same unit resolve to exactly one winner. The loser receives a claim-denied response, not a silent conflict. The platform enforces one output, one executor. The claim carries a time-to-live (TTL) that begins the lease.

Heartbeat. The agent sends heartbeats to maintain its lease. An expired lease returns the work to the pool automatically. This prevents a crashed or stalled agent from permanently locking a unit of work. The TTL is set per claim and is not negotiable by the agent.

Work report. The agent submits a work report with evidence. This is the material the Decision Owner will review at the acceptance gate. The report is a structured artifact, not free-text: it carries the results of the harness checks, the changes made, and anything the agent flagged as uncertain.

Submit for review. The agent moves the work into the human judgment queue. From this point, no agent action can advance the work further. A human must review and accept. This is the boundary between agent execution and human judgment, and the platform enforces it at the state machine level.

> *The protocol is simple by design. Available work, claim, heartbeat, report, submit. Every step is attributed, every lease expires, and every submission enters a human gate. There is no path from agent execution to accepted work that does not pass through human judgment.*

Access is layered. API keys carry channels (MCP, REST), tiers (observer, builder, author, operator, admin), and disclosure levels. The tool catalog itself is tier-filtered: an observer key does not even see mutation tools. Every tool call is access-logged with user, actor, key, and request identifier. The access model is the same whether the agent is the native runtime or an external MCP client. Governance does not vary by executor type.

The Gap the Platform Fills

The operating model described in the Singularics papers is a set of human decisions: who decides, how fast, on what evidence, with what authority. Those decisions can be made without software, and they should be understood without reference to any tool.

But running them at scale requires a mechanism. A single Decision Owner can hold the intent graph for one outcome in their head. A portfolio of twenty outcomes across fifty agents and twelve domain experts cannot be governed by memory and spreadsheets.

The platform exists for four reasons.

The graph must be explicit

When intent is implicit, alignment is a matter of opinion. When intent is explicit and connected to work through typed edges, alignment is a measurable property. The platform makes the graph explicit, navigable, and continuously updated.

The gates must be enforced

Cultural enforcement of judgment gates fails under pressure. When a deadline approaches, the first thing an organization drops is the review step. The platform makes the gates structural. They cannot be dropped, because the state machine does not have a transition that bypasses them.

The evidence trail must be automatic

A governance model that depends on people writing down what they decided is a governance model that degrades the moment people are busy. The spine produces the trail as a byproduct, which means the trail is complete even when people are under pressure.

Agent coordination must be mechanical

Coordinating agent execution across a portfolio, enforcing claim exclusivity, expiring leases, routing work reports to the right human judge, and maintaining the lifecycle state machine are tasks that do not benefit from human attention and fail when they depend on it.

> *The model needs software to run at scale. Not because the model is complicated, but because the things it requires, explicit intent, enforced gates, automatic evidence, and mechanical coordination, are precisely the things that humans do poorly under pressure and machines do reliably.*

How the Platform Connects to the Operating Model

Every element of the operating model has a concrete expression in the platform. The mapping is not coincidental. The platform was built to mechanize the model, and the model was designed to be mechanizable.

The Intent Council sees the constellation view, a projection of outcomes as a structural map of strategic intent. The three horizons (Now, Next, Later) are states in the outcome lifecycle, not separate lists. The quarterly rebalance reads the graph directly.

The Flow Council sees the signal board, a projection of the work lifecycle as lanes with the human judgment queue at the top. Weekly calibration reads Now-state outcomes against capacity. Blocker resolution, capacity release, and team formation are all spine commands with attribution and audit.

The Decision Owner works through the shaping gate (spec) and the acceptance gate (judge). Their bounded authority envelope is recorded in the platform, and escalation requirements surface automatically when an acceptance would exceed it.

The Fleet Lead sees the operations switchboard: agent telemetry, execution packets, claims, and heartbeats in real time. They decompose the daily spec into execution units and assign them to the fleet through the spine.

The Domain Knowledge Network encodes criteria into the harness and context into the knowledge graph. Both are graph records that persist beyond the expert's session and compound with every contribution.

The Harness is the control layer of the platform. It runs on every push in CI, and the review gate requires its results before the Decision Owner can accept. The harness is not a developer tool the business happens to benefit from. It is the Decision Owner's instrument.

The platform is the place where all of these elements meet. Without it, each element is a decision that must be enforced culturally. With it, each element is a constraint that the state machine enforces mechanically. The difference is the difference between a policy document and a locked door.

The decision framework for whether to adopt Intent is straightforward. If you are running three or fewer simultaneous efforts with stable teams, the operating model can be managed with lighter tooling and discipline. If you are running more than three efforts with fluid teams, agent fleets, and regulatory constraints, the coordination overhead of managing the model manually exceeds the value of the model itself. Intent exists for the second case, and the threshold is observable: when the Flow Council spends more time on governance coordination than on governance decisions, the mechanism is overdue.

The Tracker, the Spreadsheet, and the Ungoverned Orchestrator

Intent replaces three categories of tooling that exist in most organizations.

The project management tool. Tools that track work as lists, boards, and timelines without structural connection to strategic intent. Work in those tools is organized by team, by project, or by sprint. Work in Intent is organized by the intent it serves, and orphaned work is flagged as a defect.

The governance spreadsheet. Governance artifacts maintained in documents and spreadsheets that must be manually compiled and manually audited. The command spine produces the governance trail as a byproduct of normal operation. Every governance forum reads the ledger directly.

The agent orchestration layer without governance. Tools that coordinate agent execution without judgment gates, evidence requirements, or attribution. These tools make agents productive. They do not make agents governed. Intent makes

agents both, because the governance and the execution share the same state machine.

Three Mechanisms in the Control Plane

Decision ownership. The Decision Owner is the human who holds the permission gate and the acceptance gate. The platform records their bounded authority envelope and surfaces escalation requirements automatically.

The spec. The spec is what passes through the shaping gate. The platform enforces that work cannot enter execution without a spec that the shaping gate considers complete: linked intent, a description, and checkable acceptance criteria.

The harness. The harness is the control layer that produces the evidence the review gate requires. Without the harness, the review gate has no evidence to require, and acceptance degrades to demo-only judgment.