

BUSINESS EXPLAINER

Cognous Control Plane

Bounded authorization, effect-time checks and durable records for enterprise AI agents

October 2026

Before an enterprise lets an AI agent change anything, it must be able to show what was proposed, what authority was still current when the change was attempted, and what the destination was actually observed to do.

Business explainer prepared from the Cognous Control Plane whitepaper (de Lima, 2026).

1. Executive Summary

Cognous Control Plane is an open-source reference implementation for deciding whether an AI agent's proposed action is authorized, checking that authorization again immediately before the action takes effect, and keeping a durable record of what happened afterwards. It sits beside an AI agent, not inside it. It compares a declared proposal with authority supplied by a separate, trusted source, records the decision, revalidates the critical inputs at effect time, and keeps decisions, execution attempts, destination observations and reconciliation as separate facts.

It is deliberately not several things. It is not an agent framework, a model, a dashboard or a full enterprise governance suite. It is one component of the Cognous Open Control Stack, a set of 13 open components organized into six responsibility groups. The Control Plane sits in group 03, Declaration & Policy Control, alongside the Cognous Action Manifest, which declares what an agent may propose.

The central message for enterprise leaders is this:

As AI agents move from producing text to triggering refunds, updates and messages, an approval given earlier is not enough. The question is whether authority, approvals, policy and evidence were still valid at the moment the effect was attempted, and what the destination actually did.

A chatbot can be governed by reviewing its output. An agent that proposes refunds or updates systems of record cannot. For agents, the governance question is no longer "what did the model say?" but "what did the agent propose, what authority was current when it tried to act, what was attempted, and what is known about the result?"

The Control Plane is published as a **bounded synthetic reference**. Its supported workflow is one synthetic refund against a local test destination on one host. It is not a production deployment, a compliance certification or evidence that any institution has adopted it.

2. The Business Problem

AI agents are no longer just producing text. Across enterprises, they are beginning to:

- call tools and APIs;
- read enterprise data sources;
- propose payments, refunds and account changes;
- trigger workflows;
- draft and send messages;
- update systems of record;
- make recommendations that influence human decisions.

This shift creates a new class of business risk. With a chatbot, the output is the whole story. With an agent, the output is only the last step of a longer chain: a proposal, an authority check, an attempt to change something, and a result that may or may not be known. An enterprise may know that a refund was requested without knowing whether the approval behind it had been withdrawn, whether the refund system actually applied it, or whether a retry created a second one.

That gap shows up as six recurring pain points.

1. Proposals confused with permission. A tool name, a well-formed request or a fluent explanation can look like authorization. Without a separate, explicit authority check, an enterprise cannot say whether a proposed action was permitted or merely possible.

2. Stale authority. An approval, grant or policy can change between the moment a decision is made and the moment the action is attempted. A cached "yes" from earlier is not proof that the action was still allowed when it happened.

3. Missing records of refusals. When a risky action is stopped, many systems record nothing. The absence of a bad outcome is not evidence that a control worked. Risk and audit teams need held and refused actions to be visible.

4. Uncertain outcomes treated as failures. When a request times out, the change may already have happened. Treating a timeout as "nothing happened" and trying again can produce duplicate effects, such as a second refund.

5. Inconsistent decisions. When permission logic is scattered across application code, the same action under the same conditions may be handled differently in different places.

6. Runs that cannot be reconstructed. When something goes wrong, investigation often becomes guesswork across scattered logs. Enterprises need connected records of what was proposed, decided, attempted and observed.

These pain points are concrete. Consider four common deployments:

- A customer-service agent proposes a refund to a customer account.
- A workflow agent proposes an update to a production database.
- A procurement or finance agent proposes an action that requires a named approval.

- An internal agent reads records and produces a recommendation that a person acts on.

In each case, the business exposure is not primarily the model's text. It is the action layer: what the agent tried to do, under what authority, at what moment, and with what known result.

3. What Control Plane Is

Cognous Control Plane is a lightweight runtime layer that sits beside an AI agent. It does not replace the agent, the framework that orchestrates it, the model that powers it or the system that executes the action. It decides, rechecks and records.

In one sentence:

It evaluates an agent's declared proposal against independently supplied authority, checks that authority again just before the effect, and keeps decisions, attempts, observations and reconciliation as separate, durable records.

The term "control plane" is worth translating into business language. Control Plane is:

- **not the AI model itself:** it does not generate or reason;
- **not the workflow application:** it does not plan tasks or orchestrate tools;
- **not the executor:** in the Cognous stack, the Cognous Execution Runtime performs the constrained effect;
- **not a source of authority:** institutions own grants, approvals and policies; the Control Plane reads them through a trusted resolver;
- **a layer that decides and records** around agent actions, before and after those actions reach a destination.

The pattern is simple to state. An agent proposes an action that matches a declaration in an Action Manifest. A separately configured resolver supplies the current authority. The Control Plane records a decision. Immediately before the effect, it resolves the authority again and refuses if anything critical has changed. After the attempt, it keeps what the executor reported apart from what the destination was independently observed to contain.

4. What It Does

Binds decisions to an exact declared action

A proposal is tied to a specific action in a Cognous Action Manifest (version 1.1 adds the runtime bindings used here), together with its exact target, payload, adapter and requested permissions.

Business value: the decision is about one precise operation, not a loose category. Changing the target, amount or payload after approval means the earlier decision no longer applies.

Evaluates authority from a trusted source

Current authority comes from a resolver that the deploying organization configures: the grant and its status, the acting identity and any delegation, the issuer's mandate, approval records, policy versions and required evidence. In the Cognous stack, the shape of that authority follows the Authority Context profile from Cognous Institutional Governance.

Business value: permission comes from the institution's own records, not from the agent's request. A valid signature, a registry entry, a message receipt or an evidence package does not by itself authorize anything.

Revalidates at effect time

Immediately before the action reaches the destination, the Control Plane resolves the decision-critical inputs again and compares them with the recorded decision. A revoked grant, a withdrawn approval, a superseded policy or stale evidence stops the effect.

Business value: an earlier approval is not treated as a permanent credential. This is the default behavior, which the stack calls **CO**. It is a precheck, not a guarantee: a change that lands after the final check but before the destination commits can still slip through. Section 7 explains this limit.

Records held decisions

Held decisions are recorded with their reasons. (Durable records of refusals at effect time exist in newer repository source but are not part of the selected release.) In the bounded workflow, a held decision carries no authorization binding and cannot be executed.

Business value: stopped behavior becomes evidence. Risk teams can show that a specific action was proposed and why it was not allowed.

Keeps uncertain outcomes uncertain

Every action has a stable effect identity, and every execution try gets its own attempt record. If an acknowledgement is lost, the attempt is recorded as unknown rather than failed. Reconciliation checks the destination for the original effect. A fresh "not found" reading is recorded as point-in-time absence and does not by itself permit a retry.

Business value: this reduces the risk of turning a timeout into a duplicate refund. It also gives incident reviewers an honest picture: what was attempted, what was acknowledged, and what was actually observed.

Persists records safely on one host

Records are written through a store that locks each read-and-append transaction on the local Linux host, replaces files atomically and reports storage failures explicitly instead of hiding them.

Business value: cooperating processes on the same machine do not silently lose each other's records. The guarantee is local: it does not cover shared network drives, several hosts, or the whole decide, execute and reconcile sequence as one transaction.

Retains the original run-record toolkit

The first-generation tools remain available: a deterministic policy gate with allow, block and escalate results, rule-by-rule evaluation traces, blocked-action and reliance records, run records, replay bundles, validation reports, redacted exports, HMAC-signed bundles and a small command-line tool, `acp`.

Business value: teams can still produce simple, structured run records for experiments. An "allow" from this older gate is a policy-gate result, not an institutional grant; new integrations are directed to the bounded workflow.

Feeds downstream evidence

The records the Control Plane keeps are consumed by other stack components. The Cognous Replay Bundle reconstructs them without re-running anything, and the Cognous Governance Evidence Pack turns reconstructed records into review material.

Business value: auditors receive connected evidence instead of separate logs. Reconstruction is not re-authorization, and an evidence package is not independent assurance.

5. Why It Is Valuable

The benefits below describe what the design is intended to support. None of them has been measured in an enterprise deployment, and the reference makes no claim about cost savings, review time or compliance outcomes.

Clearer authority. Reviewers can see which grant, approval, policy and evidence a decision depended on, rather than inferring it from configuration files.

Current-state checks. Authority is checked again just before an action takes effect, so a withdrawn approval is designed to stop the action rather than be discovered afterwards.

Honest outcome records. Attempts, acknowledgements and destination observations are separate, so an unknown result stays unknown instead of being reported as success or failure.

Visible holds. Held decisions leave records with reasons. Durable records of refusals at effect time exist in newer repository source but are not in the selected release.

Safer recovery. Retries are not triggered merely because a destination did not report the effect at one moment.

Consistent decisions. The same proposal and the same authority inputs produce the same binding, which reduces ad hoc permission handling spread through application code.

Separation of responsibilities. Application teams keep their agent framework and model choices. The institution keeps ownership of its authority records. The control layer sits between them.

6. How It Is Different from Other Approaches

Enterprises evaluating agent governance encounter several overlapping tool categories. Control Plane is complementary to most of them, and it is important to be precise about the differences.

Versus agent frameworks

Agent frameworks help agents plan, use tools and orchestrate multi-step workflows. Control Plane does not try to be the framework. It sits beside the framework and decides whether a proposed action is authorized when it is about to take effect.

Difference: Frameworks help agents act. Control Plane decides whether a specific action is still authorized and records what happened.

Versus observability and logging tools

Logging tools record system events, typically during or after execution. They rarely distinguish a proposal from a permission, an acknowledgement from an observed result, or a timeout from a failure.

Difference: Logging says what happened. Control Plane records what was proposed, what authority was current, what was attempted and what was observed, as separate facts.

Versus agent policy engines

Policy engines and agent governance toolkits evaluate tool calls against rules. The Control Plane repository includes an optional compatibility adapter for the Microsoft Agent Governance Toolkit's `pre_tool_call` check, pinned to an exact upstream version. In that adapter, the toolkit can refuse, or through a narrow transform change the payload, which then requires a fresh Cognous authority decision; its "allow" never replaces the Cognous authority check. The adapter is merged in the repository but is not part of the selected stack release.

Difference: A policy engine answers "do the rules allow this call?" Control Plane also asks "is the institutional authority for this exact operation still current?"

Versus dashboards

Dashboards visualize data. They are only as good as the records beneath them. Control Plane has no user interface of its own by design.

Difference: A dashboard is a view. Control Plane is a decision and record layer.

Versus policy documents

Policy documents define what should happen, but generate no runtime evidence. Control Plane records which policy versions and approvals each decision depended on.

Difference: Policy documents define intent. Control Plane records how current authority was applied to each action.

Versus model safety filters

Safety filters screen generated content. That remains necessary, but it does not address which actions were attempted, under what authority, against what targets.

Difference: Safety filters govern content. Control Plane governs the action layer.

Versus full enterprise AI governance platforms

Governance platforms may cover model inventories, approval workflows, risk assessments and compliance reporting. Control Plane is intentionally narrower. It could supply evidence to such a platform, but it does not attempt to be one.

Difference: Governance platforms manage programs. Control Plane supplies the runtime decision and record layer.

Comparison at a glance

Category	What it does	What it misses	How Control Plane differs
Agent frameworks	Help agents plan, use tools, orchestrate workflows	Independent, current authority checks before effects	Sits beside the framework; decides and rechecks rather than orchestrating
Observability / logging	Records system events, often after the fact	Proposal versus permission; acknowledgement versus observed result	Keeps decisions, attempts, observations and reconciliation separate
Agent policy engines	Evaluate tool calls against rules	Institution-owned grants, approvals and evidence at effect time	Treats such engines as an extra refusal check, never as authority
Dashboards	Visualize data	The underlying structured records	Produces the records a dashboard could display
Policy documents	Define what should happen	Runtime evidence that policy was applied	Records the policy versions each decision used
Model safety filters	Screen generated content	Tool use, authority, action targets	Governs the action layer, not just the text
Enterprise governance platforms	Manage AI programs: inventories, approvals, reporting	Fine-grained, effect-time action evidence	Provides a narrow runtime layer such platforms could consume

7. What It Is Not

Clear scope is part of the design. Cognous Control Plane is not:

- a complete enterprise AI governance platform;
- an agent framework or model runtime;
- a source of institutional authority;
- a hosted dashboard;
- a production identity, credential or key-management system;
- a payment system or evidence of real settlement;
- a substitute for good policy design;
- a guarantee that model outputs are correct;
- a compliance certification;
- a complete security boundary by itself.

Several limits deserve plain statement:

- **The default check is not atomic with the destination.** C0 revalidates just before dispatch, but a change made after that check and before the destination commits can still be followed by an effect. A later qualification in the Control Plane repository, which is not part of the selected release run, deliberately demonstrates this residual race.
- **The stronger ordering is optional and local.** An optional profile, called **C1**, orders authority changes and the protected effect through one SQLite transaction on the same host. It is included in the selected source but is not enabled by default, and it covers only cooperating local writers. No cross-host or external-destination guarantee (the stack's C2 and C3 levels) is claimed.
- **Effect identity is not business intent.** The default path prevents the same effect from being applied twice, but two separately authorized operations for the same underlying refund can both execute. A separate, optional refund-intent profile in the Execution Runtime addresses that local case; the default path does not enable it.
- **Supplied authority is trusted as configured.** The reference resolver is synthetic and is not institutionally authenticated. If configured grants or policies are wrong, the Control Plane will apply them faithfully.

It should be deployed alongside application authorization, infrastructure security, identity and credential controls, model-risk controls and operational monitoring.

8. Example Scenario

Consider a customer-service agent that proposes a refund in the synthetic reference workflow.

1. **The action is declared.** The Action Manifest describes a refund action with limits on amount and number of effects. The declaration grants nothing by itself.
2. **The agent proposes a refund.** The proposal names the declared action, the target account, the amount and the payload.
3. **Authority is resolved independently.** A trusted resolver supplies the current grant, approvals, policy versions and required evidence. The incoming request does not carry authority.
4. **The Control Plane records a decision.** All checks pass, so the decision is authorized and bound to the exact operation and grant revision.
5. **The approval is withdrawn.** Before the refund is dispatched, a reviewer withdraws the approval.
6. **Effect-time revalidation refuses.** Immediately before dispatch, the Control Plane resolves authority again, finds the changed approval and stops. No refund reaches the destination. The earlier decision is not treated as reusable.
7. **A fresh proposal is authorized and dispatched.** With valid approval, a later decision passes revalidation and the executor attempts effect E1.

8. **The acknowledgement is lost.** The call times out. The attempt is recorded as unknown, not failed.
9. **Reconciliation checks E1.** The destination is observed independently. If E1 is found applied, delivery is resolved without a second dispatch, and the interrupted acknowledgement history is kept. If E1 is not found, that is recorded as point-in-time absence; it does not authorize a retry, because the original request might still commit.
10. **Records are reconstructed.** The Replay Bundle reconstructs the decisions, attempts and observations, and the Governance Evidence Pack packages them for review, without re-running the refund.

The business outcome: the enterprise can show, with structured records, what was proposed, what authority was current at each step, what was stopped, what was attempted and what the destination was observed to do. If a question arises next quarter, the answer is a record, not a reconstruction from memory. This remains a synthetic example: it does not show external payment settlement.

9. Why This Matters Now

Enterprises are moving from AI experimentation to AI deployment, and the risk profile changes at exactly that transition. A pilot agent that drafts text in a sandbox is a productivity experiment. An agent that proposes refunds, changes records and sends communications is an operational actor, and operational actors need operational controls.

As delegated authority increases, the timing of authority matters. Every additional tool an agent can call widens the gap between an earlier approval and the moment an effect actually happens.

A useful framing for leadership:

- **Chatbots require output review.** The text is the risk surface.
- **Agents require action governance.** The tool calls, data access and system changes are the risk surface.
- **Enterprise agents require current authority and honest outcome records.** Authority must be checked when an effect is attempted, and uncertain results must stay visibly uncertain, for security, audit, legal and customer review.

Organizations that design for these questions before scaling agent deployment can treat governance as an engineering pattern. Organizations that scale first may find themselves retrofitting authority checks and outcome records onto systems that were never built to produce them.

10. Open-Source Reference Implementation

Cognous Control Plane is published at github.com/cogno-us/cognous-control-plane under the Apache-2.0 license. It is part of the Cognous Open Control Stack (github.com/cogno-us/cognous-control-plane).

open-control-stack), whose component lock selects the exact component revisions used in the supported integration.

In practical terms, the repository includes:

- a Python package (agent-control-plane, built on Pydantic; package names were kept when the component was renamed) with the bounded authorization workflow, its resolver interface, durable record store and a synthetic refund destination;
- the opt-in same-host authority/effect claim profile (C1);
- a proposed, non-authorizing Decision Input Commitment record with its own verifier and test vectors;
- the original run-record toolkit, JSON schemas for its record formats and the acp command-line tool for validating, redacting, signing and verifying exported records;
- an optional Microsoft Agent Governance Toolkit compatibility adapter;
- worked examples and a test suite run in GitHub Actions on Python 3.11 and 3.12.

Getting started from a fresh checkout, with Python 3.11 or later in an activated virtual environment:

```
python -m pip install -e ".[dev]"
acp --help
acp validate-run examples/sample_run_record.json
python examples/policy_config_demo.py
```

The full selected integration is run from the stack hub, whose runner checks out the exact locked components and executes the synthetic refund workflow in two isolated repetitions.

Current status. The stack is a merged, pinned bounded synthetic reference. Its selected release evidence resolved all 35 acceptance scenarios, with required suites executed twice. A fresh reproduction on 13 October 2026 ran 1,009 Python test cases in each of two repetitions plus 20 Node tests; 34 checked invariants passed and one known limitation, the business-intent duplicate, was characterized rather than prevented. These are whole-stack figures, not Control Plane counts. The optional C1 profile passed 73 tests in a separate run that is not added to that total. Feature intake was frozen for release closeout on 13 October; public-preview approval is pending, and final publication and operational trust remain on hold.

The repository has continued to evolve after the revision the stack currently selects. Tenant-bound authorization, durable records of effect-time refusals, records of each authority input actually read, stricter handling of ambiguous dispatch errors and the policy-toolkit adapter are merged in the repository but are not part of the selected release.

The implementation is intentionally small. It is meant to be readable and verifiable by an enterprise platform team: a pattern to evaluate and build on, not a black box to trust.

11. Strategic Summary

Cognous Control Plane gives enterprises a concrete reference for adding current-authority checks and honest outcome records to AI-agent workflows without replacing their agent framework or model stack. It binds each decision to an exact declared action, takes authority from a trusted institutional

source, rechecks that authority immediately before the effect, records refusals, and keeps attempts, observations and reconciliation apart so that uncertainty is not mistaken for success or failure.

It does not claim to solve AI governance. It is explicit about its limits: a residual check-to-commit race on the default path, an optional same-host stronger profile, no distributed or external-settlement guarantee, and no deduplication of business intent by default. Its contribution is narrower and more useful: making agent actions decidable, checkable and reviewable at the layer where agents actually touch enterprise systems.

Before enterprises delegate meaningful work to AI agents, they need to know what those agents proposed, what authority was current when they acted, what was stopped, what was attempted and what actually happened. That is the role of the Cognous Control Plane.