BUSINESS COLLATERAL

Cognous Open Control Stack

Open-source reference infrastructure for governable AI agents

July 2026

> *Before enterprises can scale AI agents responsibly, they need more than prompts, logs, dashboards, and policy documents. They need a structured lifecycle: declare what agents may propose, control what they actually propose, replay what happened, and evidence the deployment for governance review.*

1. Executive Summary

Cognous Open Control Stack is open-source reference infrastructure for governable AI agents. It gives enterprises a structured way to move from AI-agent demos to governed deployment by defining four linked artifacts, each answering a question that agent governance depends on:

> *Declare → Control → Replay → Evidence*

- **Declare.** The Agent Action Manifest declares what an agent may propose before runtime — its tools, actions, authority requirements, review requirements, and payload sensitivities.
- **Control.** The Agent Control Plane records and governs what the agent actually proposes at runtime — action proposals, policy decisions, blocked actions, authority context, and source reliance.
- **Replay.** The Agent Replay Bundle packages a run into a portable technical record that can be reconstructed, validated, redacted, and shared.
- **Evidence.** The Agent Governance Evidence Pack translates runtime and replay records into business-facing evidence packages for deployment review.

The stack is deliberately limited in what it claims to be. It does not run the model. It does not replace agent frameworks. It does not certify compliance, and it does not solve all of AI governance. What it supplies is the missing artifact chain between agent behavior and enterprise review — the structured records that connect what an agent was allowed to propose, what it actually proposed, what was blocked, what it relied on, and what evidence supports the deployment decision.

As enterprises move from chatbots to agents that use tools, access systems, draft communications, propose changes, and trigger workflows, governance must move from informal review to structured evidence. The Open Control Stack provides the declaration, runtime control, replay, and evidence layers needed to make agent behavior inspectable.

A chatbot can be governed by reviewing what it said. An agent must be governed by reviewing what it can do, what it tried to do, and what stood behind its output. The Open Control Stack makes those questions answerable with records rather than reconstruction.

For enterprise leaders, the Open Control Stack is not another agent framework. It is the evidence layer around agent deployment: the artifact chain security, risk, audit, legal, and platform teams need before agents are allowed to touch business systems at scale.

2. The Business Problem

Across enterprises, AI agents are beginning to:

- call tools and APIs;
- access customer, financial, operational, and internal data;
- summarize records and files;
- draft and send communications;
- propose database updates;
- approve or escalate workflow steps;
- prepare exports;
- make recommendations that influence business decisions.

This shift creates a class of exposure that existing governance machinery was not designed for. Most enterprise governance systems were built for software, workflows, models, policies, or human approvals — not for autonomous or semi-autonomous agents that propose actions through tools. With a chatbot, the output is the whole story. With an agent, the output is only the last step of a longer chain of proposed, permitted, blocked, and executed actions.

In practice, the gap shows up in four places.

1. The declaration gap. The enterprise may know the agent's purpose — "it handles customer inquiries," "it drafts purchase orders" — but not its full action surface: what actions it may propose, through which tools, under what authority, with what review posture. That information usually exists, but scattered across prompts, code, tool wrappers, and institutional memory. Nobody can review it as a whole, because it does not exist as a whole.

2. The runtime control gap. The agent proposes actions during execution, but ordinary logs often fail to capture the proposal, the policy decision, the authority context, the blocked action,

and the source reliance as governance records. The absence of a bad outcome is not evidence that a control worked.

3. The replay gap. After an incident or audit question, the enterprise may not be able to reconstruct a specific run from scattered logs and traces. Investigation becomes forensic guesswork organized around systems, when the reviewer's question is organized around the run.

4. The evidence gap. Business reviewers may receive technical records but no coherent evidence package showing purpose, exposure, controls, blocked behavior, risks, validation, redaction, and review decisions. Risk officers, auditors, legal teams, and executive sponsors do not read runtime traces. They read evidence.

Each gap could be patched in isolation. But governed AI-agent deployment needs an artifact chain, not isolated tooling. That is why the Open Control Stack exists: the four layers are designed to feed one another, so a declaration made before deployment becomes a baseline at runtime, runtime records become a replayable bundle, and replay evidence becomes reviewable business material.

3. The Stack in One View

Layer	Artifact	Question answered	Primary audience	Output
Declare	Agent Action Manifest	What may this agent propose?	Security, governance, platform teams	Declared action surface
Control	Agent Control Plane	What did the agent propose, and what did policy decide?	Engineering, security, runtime governance	Runtime control records
Replay	Agent Replay Bundle	Can we reconstruct this run?	Audit, incident response, platform teams	Portable replay record
Evidence	Agent Governance Evidence Pack	What evidence supports this deployment review?	Executives, risk, compliance, audit, legal	Business-facing evidence package

The stack is valuable because each layer feeds the next. Declaration gives runtime control a baseline: policy can be evaluated against a declared action surface rather than improvised permission logic. Runtime control generates the records — proposals, decisions, blocks, authority, reliance — that no later layer can invent after the fact. Replay bundles package those records into portable, run-level artifacts. Evidence packs translate those artifacts into business-facing review material. Remove any layer and the chain weakens: undeclared surfaces cannot be checked, unrecorded proposals cannot be replayed, and unpackaged records cannot be reviewed.

4. Layer 1 — Declare: Agent Action Manifest

Agent Action Manifest is the pre-runtime declaration layer of the stack. It creates a structured action inventory before the agent runs — a reviewable, portable statement of the agent's action surface, produced before the agent touches enterprise systems.

> *It gives enterprises a structured answer to the question: "What is this agent allowed to propose?"*

The manifest declares:

- the tools the agent can use;
- the specific actions it may propose through those tools;
- the type of each action — reads, writes, sends, exports, approvals do not carry the same risk;
- which actions require authority;
- which actions require human review;
- which actions should produce reliance evidence;
- payload policies identifying the fields that matter;
- redaction hints marking fields that should be masked in downstream records;
- validation results confirming the declaration is internally consistent.

The distinction between tool access and action semantics is the heart of the manifest. A tool list shows what systems an agent can call, but not what it will do with them. Access to a CRM does not distinguish reading a record from updating one, and access to an email system does not distinguish drafting a message from sending it. The manifest makes those distinctions explicit — which is what allows privileged actions to be identified before deployment, review requirements to be set deliberately, and out-of-scope behavior to be detected later.

Example. A customer-service agent may read CRM records, search account notes, draft emails, and send emails. The manifest declares those as four separate actions with different risk profiles and different review requirements — the send action carrying authority and review requirements that the read actions do not.

What it is not. The manifest does not run the agent, enforce policy, grant authority, or guarantee review. It declares the expected action surface — and gives every downstream layer, and every downstream reviewer, a baseline to check against.

5. Layer 2 — Control: Agent Control Plane

Agent Control Plane is the runtime layer of the stack. It sits beside the agent — not inside it — and records and governs proposed actions before those actions are treated as executable. It does not replace the agent framework or the model; it records and governs the action layer around them.

> *It gives enterprises a structured record of what the agent tried to do, what was allowed, what was blocked, why policy decided that way, what sources were relied on, and how the run can be replayed.*

At runtime, the control plane records:

- action proposals, captured before execution — which tool, what type of action, what target;
- policy decisions, produced by a deterministic policy gate so identical inputs produce identical decisions;
- blocked actions, preserved as first-class records rather than silent non-events;
- authority context — what permission existed for that run, scoped by actor and action type;
- reliance records — the tools, files, databases, and inputs the agent depended on;
- policy evaluation traces showing which rules were checked, which matched, and why the result followed;
- run records assembling the full sequence;
- replay bundles packaged at the end of a run;
- redacted exports that preserve structure while masking sensitive content;
- signed replay bundles carrying export-integrity metadata.

The business value is in the conversion: proposed agent behavior becomes records. The enterprise can distinguish what the agent wanted to do from what it actually did — the foundation of meaningful agent oversight. Blocked actions become visible evidence that controls operated. Policy decisions become consistent and traceable rather than scattered through application code. Authority and reliance context is preserved at the moment of decision, when it is cheap to capture, rather than reconstructed later, when it is expensive or impossible.

Example. If a customer-service agent proposes sending an email without outbound-send authority, the control plane records the proposal, blocks it, records the reason — outbound-send authority missing, after the tool itself passed the allowed-tool check — and preserves the block as governance evidence.

What it is not. The control plane is not an agent framework, a model runtime, a dashboard, or a complete governance platform, and it is not a substitute for application authorization. It is the record and control layer those tools generally do not provide.

6. Layer 3 — Replay: Agent Replay Bundle

Agent Replay Bundle is the run reconstruction layer of the stack. It packages the technical evidence of a run into one portable record, organized around the question a reviewer actually asks: what happened here, and can we see it?

> *It gives enterprises a structured answer to the question: "Can we reconstruct this agent run and review what happened?"*

A replay bundle preserves, in one artifact:

- the run frame — the context in which the run occurred;
- the action proposals the agent made;
- the policy decisions made in response;
- the policy evaluation traces behind those decisions;
- the actions that were blocked;
- the authority records that existed for the run;
- the reliance records showing what the agent depended on;
- the final output;
- a validation report confirming the record's internal consistency;
- redaction metadata enabling safer sharing;
- signature metadata supporting export integrity.

The distinction from logging is worth stating plainly: a log records events. A replay bundle reconstructs the agent run. A log line may show that a function was called; it does not preserve the agent-level story — what was proposed, decided, blocked, permitted, and relied on — as one coherent, portable package. Replay bundles make run evidence portable across teams and systems, keep blocked actions alive as first-class evidence, support redaction so records can be shared without exposing unnecessary sensitive data, and carry signing metadata so a reviewer can detect whether a bundle was modified after it left the runtime environment. Incident review becomes record-based rather than reconstruction-based.

What it is not. A replay bundle does not run the agent, enforce policy, certify compliance, or prove that the agent was correct, and the reference implementation does not provide production key management. It is the technical evidence artifact that review processes can inspect.

7. Layer 4 — Evidence: Agent Governance Evidence Pack

Agent Governance Evidence Pack is the business-facing evidence layer of the stack. It translates runtime and replay evidence into a structured package organized around deployment review rather than around technical events.

> *It gives enterprises a structured answer to the question: "What evidence do we have that this agent deployment is governed?"*

An evidence pack summarizes, in one reviewable artifact:

- the agent overview — what the agent is deployed to do;
- the deployment context — where it runs and which systems it touches;
- the tool inventory and action inventory;
- the authority model governing privileged actions;
- the policy controls in place;
- blocked-action summaries showing that controls operated;
- reliance summaries showing what sources the agent depended on;
- the inventory of available replay bundles;
- validation results across the underlying records;
- the redaction and export posture, including signed exports;
- the risk register, with open items visible rather than buried;
- review records — who reviewed the evidence and what was decided.

The value is translation. Runtime records are necessary but structured around events, identifiers, and payloads — they serve developers and incident responders well, but executives, auditors, risk teams, and business owners need to understand purpose, controls, risks, and decisions. The evidence pack turns scattered logs, bundles, tickets, spreadsheets, and notes into one artifact that connects controls, risks, evidence, and review decisions — supporting governance review and deployment discipline across the functions that must sign off.

What it is not. An evidence pack does not run agents, enforce policy, or certify compliance, and it does not replace security, audit, legal, or governance processes. It creates the structured evidence artifact those processes can inspect.

8. How the Four Layers Work Together

The stack is easiest to understand through a realistic deployment. Suppose an enterprise is preparing to deploy a customer-service agent that reviews accounts and drafts customer communications.

1. Declare. Before deployment, the team creates an Agent Action Manifest. It declares four actions: CRM record read, account-note search, email draft, and email send. It marks email send as an external send requiring explicit authority and human review. It marks customer identifiers and message bodies as sensitive payload fields with redaction hints. Security and governance reviewers now have a structured declaration to assess — not a prompt and a tool list.

2. Control. At runtime, the Agent Control Plane records each proposed action against that declared baseline. The CRM read is proposed, evaluated, and allowed. The account-note search is allowed and its sources are captured as reliance records. The agent then proposes an email send — but outbound-send authority is missing for this run, so the policy gate blocks the action and records the proposal, the decision, the evaluation trace, and the reason. The block is preserved as evidence, not lost as a non-event.

3. Replay. At the end of the run, the Agent Replay Bundle packages everything: the task frame, the action proposals, the policy decisions and traces, the blocked email send, the authority context, the reliance sources, the final output, the validation result, the redaction metadata, and the signature metadata. Weeks later, when a reviewer asks what happened in that run, the answer is an artifact — not an archaeology project across logs.

4. Evidence. For the quarterly deployment review, the Agent Governance Evidence Pack summarizes the deployment: what the agent is for, which systems it touches, its declared tools and actions, the controls in place, the blocked-send history, the reliance sources it typically depends on, the available replay bundles, the validation status, the redacted exports produced for external review, the open risks, and the review decision. Risk, audit, legal, and the executive sponsor review one structured package instead of assembling the story by hand.

The business outcome: the enterprise can explain what the agent was allowed to propose, what it actually proposed, what was blocked and why, what it relied on, how any run can be reconstructed, and what evidence supports the deployment decision. Every one of those answers is a record, not a recollection.

9. What Makes the Integrated Stack Valuable

Deployment readiness. The manifest gives reviewers a structured declaration before production. Deployment review starts from a declared action surface instead of a purpose statement.

Runtime governance. The control plane records and governs proposed actions while they occur, converting agent behavior into governance records at the moment of decision.

Incident reconstruction. Replay bundles preserve the run as a portable artifact. Investigation becomes record-based rather than forensic guesswork.

Business review. Evidence packs translate technical records into executive- and audit-readable evidence, organized around the questions reviewers actually ask.

Cross-functional alignment. Engineering, security, risk, compliance, legal, audit, and business teams work from a shared artifact chain instead of competing partial views.

Audit and assurance preparation. The stack produces records that can support audit preparation, customer assurance, legal review, risk review, and governance decisions — generated from the first run onward, not reconstructed under pressure.

Blocked-action visibility. The stack treats blocked actions as evidence that controls operated. A well-governed deployment can show what was stopped, not merely that nothing bad happened.

Source reliance visibility. The stack records and summarizes what sources agents relied on, so reviewers can distinguish source-grounded outputs from unsupported ones.

Separation of responsibilities. Agent teams keep their frameworks and models. Governance teams get structured artifacts and records. Governance requirements do not force a rebuild of the agent stack.

10. How It Differs from Adjacent Categories

Enterprises evaluating agent governance encounter several overlapping categories. The Open Control Stack is complementary to most of them; the differences are worth stating precisely.

Category	What it does	What it misses	How the Open Control Stack differs
Agent frameworks	Help agents plan, use tools, and orchestrate workflows	Structured governance records of proposed and blocked actions	Sits beside the framework; declares, records, and gates action rather than orchestrating it
Prompts	Describe intended behavior in natural language	No structured, validated inventory of actions, authority, or review requirements	Replaces prompt-as-declaration with a validated action manifest
Tool lists	Show which systems an agent can call	Action semantics — reads, writes, sends, and approvals carry different risk	Declares actions and their types, not just tool access
Logs and observability	Record system events during or after execution	Proposals, policy decisions, blocks, authority, and reliance as governance records	Records the action layer as first-class evidence, organized around the run
Dashboards	Visualize whatever data exists beneath them	The structured records themselves	Creates the records a dashboard could later display
Policy documents	State what should happen	Runtime evidence of what policy actually decided	Records policy decisions action by action, run by run
GRC checklists	Structure periodic governance and compliance review	Machine-validated, deployment-specific artifacts tied to runtime records	Supplies the artifact chain a checklist review can inspect
AI governance platforms	Manage program-level inventories, workflows, and reporting	The run-level declaration, control, and replay artifacts	Intentionally narrower; produces evidence a platform could consume
ODES	Proposed public standard for portable decision evidence	Not an implementation; does not supply agent runtime records	Reference implementation family; may produce evidence that could support ODES-style records in future profiles

The ODES relationship deserves careful language. ODES is a separate proposed open, vendor-neutral public standard for portable decision evidence in AI-mediated and cross-boundary decisions. The Open Control Stack is not ODES and does not define ODES conformance. It may produce, package, or reference evidence that could support ODES-style records in future profiles. In short:

- **ODES** = proposed public standard for portable decision evidence;
- **Cognous Open Control Stack** = open-source reference implementation family for governed AI-agent workflows;
- **Relationship** = optional mapping and supporting evidence — not identity, and not conformance.

11. What the Stack Is Not

Scoped claims are part of the design. Cognous Open Control Stack is not:

- an agent framework;
- a model runtime;
- a hosted dashboard;
- a complete enterprise AI governance platform;
- a compliance certification product;
- a security boundary by itself;
- a substitute for application authorization;
- a guarantee that model outputs are correct;
- a guarantee that regulatory obligations have been satisfied;
- the ODES standard;
- a proprietary trust network.

Its value is focused. It supplies structured artifacts for declaration, runtime control, replay, and evidence. The quality of the resulting records depends on upstream systems, policy design, authorization, monitoring, security controls, governance processes, and implementation discipline. The stack makes agent behavior inspectable; it does not make the surrounding program sound on its own.

12. Why Open Source Matters

The stack is published as open-source reference infrastructure so enterprise teams can inspect the formats, run the examples, validate records, adapt schemas, and build compatible workflows before committing to any platform. For artifacts whose entire purpose is trust and review, inspectability is not a distribution choice — it is a design requirement.

Publishing the reference implementation openly provides:

- **Transparency.** Reviewers can see exactly what a manifest, control record, replay bundle, or evidence pack contains.
- **Vendor-neutral evaluation.** Teams can assess the artifact chain on its merits, without a procurement commitment.
- **Integration clarity.** Published, inspectable schemas and examples make it clear how other systems can produce and consume the artifacts.
- **Category definition.** Open reference artifacts help establish what the declaration, control, replay, and evidence layers should contain — a shared vocabulary across teams and vendors.
- **Lower adoption friction.** Platform teams can pilot the artifact chain in a bounded way before any broader commitment.
- **Compatibility.** The artifacts are designed to feed broader governance systems rather than compete with them.

The open-source layer is intentionally narrow: formats, validators, reference behavior, and examples, readable and verifiable by an enterprise platform team. Commercial systems can compete above that layer — on hosted services, integrations, user interfaces, approval workflows, compliance mappings, enterprise storage, identity integration, analytics, support, and operational depth. The reference implementation defines the artifacts; it does not attempt to be the platform.

13. Why This Matters Now

Enterprises are moving from AI experimentation to AI-agent deployment. The risk surface changes materially when agents gain access to tools, data, systems, workflows, and communication channels: exposure shifts from what the model says to what the agent does. The governance progression follows the capability progression:

- Chatbots require output review.
- Agents require action governance.
- Governed agents require replayable records.
- Enterprise agent programs require business-facing evidence.

The timing question is practical, not rhetorical. Organizations that establish the artifact chain before scaling will be better positioned for audit, incident response, legal review, security signoff, customer assurance, and executive oversight — because the evidence exists from the first governed run onward. Organizations that scale first will reconstruct evidence after the fact, usually under pressure, usually in response to a question they can no longer answer cleanly. The cost difference between recording at the moment of decision and reconstructing after an incident is the core economic argument for adopting the stack early.

14. Strategic Summary

Cognous Open Control Stack provides a practical artifact chain for governed AI-agent deployment:

- **Declare** what the agent may propose.
- **Control** what it actually proposes.
- **Replay** what happened.
- **Evidence** the deployment for governance review.

It does not claim to solve AI governance wholesale. It claims something narrower and more operational: governed agents need structured records at the action layer, and those records need to move through a lifecycle — from declaration, to runtime control, to replay, to business-facing evidence. Each layer answers a question the next layer depends on, and together they turn agent governance from an assertion into an inspectable record.

> *Before enterprises can scale governed AI agents, they need more than prompts, logs, and dashboards. They need an open control stack.*