

Cognous Governance Evidence Pack: Traceable, Business-Facing Evidence for Governed AI-Agent Deployment

André de Lima
Cognous
Seattle, Washington
andredelima@cogno.us

October 2026

Abstract

Enterprise AI agents increasingly use tools, retrieve data, propose actions, rely on external sources, trigger workflows and produce outputs that affect business decisions. As organizations move from demonstrations toward governed deployment, technical runtime records alone are not sufficient. Security, risk, compliance, audit, legal and executive reviewers need business-facing evidence that explains what an agent is deployed to do, which systems it can access, which actions it can propose, which authority and review controls apply, what was blocked, what sources were relied on, which reconstruction records exist, what validation found, what redacted or signed exports were produced, what risks remain open and who reviewed the evidence. They also need to know how strong each piece of that evidence is.

Cognous Governance Evidence Pack (formerly Cognous Agent Governance Evidence Pack) is a public reference format, validator, traceable importer and Markdown renderer for translating AI-agent governance records into business-facing evidence packages. It provides JSON Schema files, a Pydantic object model, a semantic validator, a deterministic importer from Action Manifest 1.1 and Reconstruction Bundle 0.2.0 artifacts, standard and traceable Markdown renderers, command-line utilities, examples and qualification workflows. Imported packs record exact input digests, producer revisions and transformation versions; keep authorization, acknowledgement, destination observation, retry permission and independent verification as separate lifecycle facts; and grade control evidence on separate levels from *declared* to *independently audited*. The component is intentionally scoped: it is not an agent framework, a model runtime, a source of authority, a compliance certification product, a hosted dashboard, a security boundary by itself or a complete governance platform. Within the Cognous Open Control Stack it belongs to the Replay & Evidence responsibility group, and its selected revision is qualified only as part of a bounded synthetic reference.

1 Introduction

AI-agent systems are moving from isolated prototypes toward operational deployment. In enterprise settings, agents may read customer records, search internal documents, draft communications, generate recommendations, propose updates to systems of record and participate in workflows that require authority or review. These capabilities create a governance problem that cannot be solved by technical logs alone.

A runtime system may produce detailed records: action proposals, authorization decisions, execution attempts, destination observations, reconstruction bundles, validation reports, redaction metadata and integrity commitments. These records are necessary, but they are not sufficient for business review. A secu-

rity reviewer, audit committee, risk officer, customer-facing compliance team or executive sponsor needs a higher-level artifact that answers the business questions around a deployment.

Those questions include:

1. What agent is being reviewed?
2. What business purpose does the agent serve?
3. Where is it deployed?
4. What systems, tools and data domains does it touch?
5. What actions can it propose?
6. Which actions require authority?
7. Which actions require human review?
8. What policy controls are declared, and which are evidenced beyond declaration?
9. What actions were blocked, held or denied?
10. What sources did the agent rely on?
11. What reconstruction bundles or run records exist?
12. What validation results are available?
13. What remains unknown or unresolved?
14. What exports were redacted or signed?
15. What risks remain open?
16. Who reviewed the evidence, and what decision was reached?

Governance Evidence Pack addresses this evidence-translation problem. It converts technical runtime and governance records into a structured, business-readable package that can be validated for internal consistency, rendered as Markdown and used as a review artifact. Since the first edition of this paper (July 2026) the component has acquired a traceable import path. A pack can now be generated deterministically from a declared Action Manifest and a retained Replay reconstruction, with provenance sufficient for a reviewer to follow each summarized fact back to its source record and producer revision.

The guiding design principle is now stated in two parts:

No enterprise agent should move from demonstration to governed deployment without a business-facing evidence package that summarizes its purpose, action surface, controls, runtime evidence, risks and review decisions. And no such package should state its evidence more strongly than its sources support.

The public reference implementation exposes sixteen public model objects:

EvidencePack	AgentOverview	DeploymentContext
ToolInventoryItem	ActionInventoryItem	AuthorityModelSummary
PolicyControlSummary	BlockedActionSummary	RelianceSummary
ReplayBundleInventoryItem	ValidationSummary	RedactionExportSummary
RiskRegisterItem	ReviewRecord	ValidationIssue
ValidationReport		

It also exposes loading, dumping, validation, summary and rendering helpers; three import functions (`import_manifest_reconstruction` and two builders that accept artifacts or file paths); a traceable renderer; the `agep` command-line interface; JSON schemas; examples; and test and qualification suites. Package, module and command names are unchanged by the repository rename:

- distribution: `agent-governance-evidence-pack`;
- Python module: `agent_governance_evidence_pack`;
- command: `agep`.

2 Problem Statement

Enterprise AI-agent governance has a translation gap. Runtime systems may produce technical records, but business reviewers need evidence packages organized around deployment questions, risk controls and review decisions. The gap appears in several operational forms.

2.1 Runtime Records Are Too Technical for Business Review

Technical records are structured around events, identifiers, traces and payloads. They suit developers and incident responders but not executives, risk teams, auditors or business owners. A reconstruction bundle may describe what was recorded in a run, but a governance review also needs to understand what the deployment is for, what controls exist, what risks remain and what decision was made.

2.2 Evidence Is Scattered Across Systems

Relevant evidence may live in action manifests, control-plane records, executor records, reconstruction bundles, validation reports, redacted exports, risk registers, approval workflows, monitoring systems and manual review notes. Without a common evidence package, reviewers must assemble the story manually.

2.3 Governance Reviews Lack a Stable Artifact

Many organizations conduct AI reviews through meetings, spreadsheets, slide decks, ticket comments or policy checklists. These are rarely structured enough to validate, render consistently or compare across deployments. Agentic systems require review artifacts that can be produced, versioned, inspected and updated as deployment posture changes.

2.4 Blocked and Unresolved Behavior Is Often Invisible

A successful deployment is not only one where harmful actions did not occur. It is one where inappropriate or unauthorized actions were proposed and stopped, and where uncertain outcomes were recorded as uncertain. Reviewers need explicit summaries of blocked actions and explicit statements of what is unknown, for example an execution whose acknowledgement was lost and whose destination state has not yet been observed.

2.5 Source Reliance Is Hard to Summarize

Agents rely on documents, APIs, databases, tools, user inputs and model outputs. Technical reliance records may exist, but reviewers need an aggregate: which sources were relied on, how often, for what scope, and where the supporting records can be found.

2.6 Validation and Export Integrity Are Not Business-Visible

Validation reports, redaction metadata, digests and signatures are technical mechanisms. Their business relevance is that they indicate whether evidence is internally consistent, whether sensitive material was removed before sharing and whether records can be checked for modification. These signals need to be visible in a business-facing artifact, together with their limits: a shared-secret integrity check, for example, does not establish public issuer identity.

2.7 Risk and Review Decisions Are Detached from Runtime Evidence

Risk registers and review decisions often exist separately from runtime evidence. An evidence pack connects review posture to the underlying records: what risks remain open, what controls support the deployment, what reconstruction bundles exist and who approved, rejected or conditionally approved the deployment.

2.8 Summaries Inflate Assurance

The first edition of this paper did not name the most consequential failure of evidence summaries: they tend to strengthen claims as they compress them. A declared control is reported as implemented; import-time validation is reported as testing; an unknown acknowledgement is reported as failure; a missing observation is counted as zero; an earlier authorization is read as current permission; a complete reconstruction is read as a completed effect. Each step makes the summary easier to read and less accurate. A business-facing evidence format must be designed to resist this drift rather than merely to compress.

3 Design Goals

Governance Evidence Pack is designed around twelve goals.

1. **Business-facing evidence.** Governance evidence should be summarized in a form business stakeholders can inspect.
2. **Deployment context.** The package should identify the agent, business purpose, environment, systems touched, data domains and ownership.
3. **Action inventory.** The package should summarize what actions the agent can propose and whether they require authority, review or reliance records.
4. **Control summary.** Authority models and policy controls should be represented explicitly.
5. **Blocked-action visibility.** Blocked behavior should be summarized as evidence, not omitted.
6. **Reliance visibility.** Sources used by the agent should be summarized for review.
7. **Replay and validation linkage.** Reconstruction bundles and validation summaries should be referenced in the package.
8. **Redaction and export tracking.** Redacted exports and intended recipients should be visible, and redacted derivatives should remain distinct from originals.
9. **Risk and review integration.** Risks and governance decisions should be captured in the same artifact as runtime evidence.
10. **Portable rendering and validation.** Packs should be loadable, inspectable, validated and rendered through a small public schema and CLI.

11. **Traceable derivation.** Generated content should record its inputs, their digests, the producer revisions that created them and the transformation that produced it, so that a reviewer can trace each summary to its source.
12. **Assurance separation.** Declared, implemented, validated, tested, observed and independently audited evidence should remain distinguishable, and missing, unknown, unavailable and redacted states should never be normalized to success or zero.

4 Non-Goals

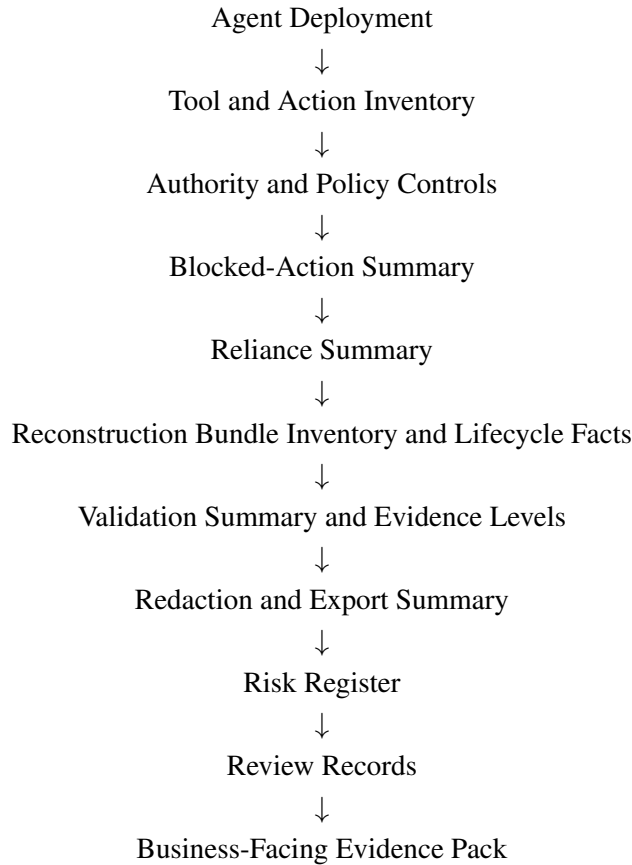
The architecture is deliberately bounded. Governance Evidence Pack is not:

- an agent framework;
- a model runtime;
- a runtime control plane or enforcement layer;
- a source of authority, permission or retry eligibility;
- a replay viewer;
- a hosted dashboard;
- a compliance certification product, legal advice or an independent audit;
- a complete enterprise governance platform;
- a production authorization system;
- a security boundary by itself;
- a substitute for application-level authorization;
- a verifier of external settlement or other real-world effects;
- a guarantee that model outputs are true, safe or complete;
- a guarantee that regulatory obligations have been satisfied.

It summarizes available governance evidence and validates package consistency. It does not execute agents, enforce runtime access control, certify compliance or prove that a deployment is safe. A valid pack, a matching digest, a signature on an input, a chain inclusion, a message receipt or a reasoning instruction never authorizes execution.

5 System Overview

The evidence packaging pattern is unchanged in outline:



An evidence pack is created during deployment review, governance review, incident review or periodic audit. It may be authored manually, assembled from other governance tooling or, where supported producers exist, generated by the traceable importer.

5.1 Position in the Cognous Open Control Stack

The first edition placed the component at the end of a four-layer sequence: Declare, Control, Replay, Evidence. That description is superseded. The Cognous Open Control Stack now comprises 13 components organized into six responsibility groups, with Governed Exchange (GAX/IMX) as a cross-cutting integration:

Group	Components
01 Knowledge & Provenance	Evidence Registry (The Index); Evidence Attestation (BitRep)
02 Institutional Governance	Institutional Governance (Alvorada)
03 Declaration & Policy Control	Action Manifest; Control Plane
04 Constrained Execution	Execution Runtime (Moltbot Safe)
05 Replay & Evidence	Replay Bundle; Governance Evidence Pack; ODES
06 Reasoning & Human Oversight	PRP; Research Intelligence; TFA
Cross-cutting	Governed Exchange (GAX/IMX)

The groups form a responsibility map, not an execution sequence or an installation requirement. Within group 05, the three components divide the evidence work:

- **Replay Bundle** imports retained Control Plane and executor records from exact supported producer revisions into a Reconstruction Bundle (format 0.2.0). It reconstructs records; it does not rerun execution, re-authorize a request or verify the destination independently.
- **Governance Evidence Pack** combines that reconstruction with the declared Action Manifest (version 1.1, which adds deterministic runtime bindings) and translates both into a business-facing review package.
- **ODES** exports decision-evidence packages that an external recipient validates and evaluates under its own policy.

The stack supports one workflow: a synthetic refund. A Manifest declares the refund action; a separately configured resolver supplies Authority Context; the Control Plane decides and revalidates decision-critical inputs before the constrained executor reaches a synthetic SQLite destination. In the reference run, the hub consumes the original retained Replay, ODES and IMX artifacts for the transported operation and produces a Governance Evidence Pack, checking effect content and count, decision/effect/attempt continuity, original artifact commitments and recipient assurance limits. The Evidence Pack and ODES identifiers name derived artifacts; they do not mint an effect.

6 Core Object Model

The object model has not changed since the first edition; the schema files and Pydantic models are the same public boundary. What has changed is that imported packs carry a structured provenance object in `metadata` (Section 17).

6.1 EvidencePack

An `EvidencePack` is the top-level object.

Field	Meaning
pack_id	unique evidence pack identifier
pack_version	format version (default 0.1; imported packs use 0.2.0)
title	human-readable title
generated_at	timestamp when the pack was generated
review_status	draft, in review, approved, approved with conditions, rejected or retired
agent_overview	agent identity and business purpose
deployment_context	environment, systems touched, data domains, geography and notes
tool_inventory	tools and systems available to the agent
action_inventory	actions the agent can propose
authority_model	summary of authority governance
policy_controls	controls applied to the deployment
blocked_actions	aggregate blocked-action records
reliance_summary	aggregate source reliance records
replay_bundles	available reconstruction bundles or run records
validation_summary	validation results across supporting records
redaction_exports	redacted or exported review artifacts
risk_register	risks, severity, status, mitigation and owner
review_records	reviewer decisions and notes
metadata	extension metadata; for imported packs, <code>traceable_import</code>

6.2 Component Objects

Each component object records the following fields.

AgentOverview agent name, optional description, business purpose, owner, business unit, user population and lifecycle stage.

DeploymentContext environment, deployment name, deployment date, systems touched, data domains, geographic scope and notes.

ToolInventoryItem tool name, description, external system, data classification, access mode and control status.

ActionInventoryItem action name, tool name, action type, description, whether authority, review or reliance is required, default posture and control status.

AuthorityModelSummary narrative summary, authority scopes, privileged action types, whether grants must expire, whether human approval is required and notes.

PolicyControlSummary control name, description, implementation status, evidence reference and notes.

BlockedActionSummary action name, type, tool, count, reason summary and evidence reference.

RelianceSummary source name, source type, count, scope summary and evidence reference.

ReplayBundleInventoryItem bundle ID, run ID, status, generation timestamp, signed flag, redacted flag, validation status and evidence reference.

ValidationSummary counts of valid and invalid bundles, warnings and errors, and an optional narrative.

RedactionExportSummary export ID, export type, redacted fields, generation timestamp, intended recipient and evidence reference.

RiskRegisterItem risk ID, title, description, severity, status, mitigation, owner and evidence reference.

ReviewRecord reviewer, reviewer role, review timestamp, decision and notes.

6.3 ValidationIssue and ValidationReport

ValidationIssue records an error or warning discovered during validation, with a stable code, a business-readable message and a field path. A **ValidationReport** aggregates issues and marks the pack valid only when no error-severity issue is present.

7 Review Status

Review status	Meaning
<code>draft</code>	evidence pack is being prepared
<code>in_review</code>	evidence pack is under review
<code>approved</code>	deployment or evidence package has been approved
<code>approved_with_conditions</code>	approval is conditional on stated review notes or mitigations
<code>rejected</code>	deployment or evidence package has been rejected
<code>retired</code>	evidence pack refers to a retired deployment

Review status is not a compliance certification. It is an evidence-package status that helps teams track review posture. The importer always emits `draft`: an attributable human review record must be supplied before any other status is meaningful, and a software-generated pack cannot create human approval.

8 Deployment Environments

The public model supports the environments `development`, `staging`, `production`, `sandbox` and `other`. Production packs receive additional warnings when review posture, redaction exports or signed reconstruction bundles are missing, reflecting the stronger evidence expectations that apply once agents move beyond experimental environments. Imported packs use `other`, because the retained producer artifacts do not establish a deployment environment; in the reference examples the producers are synthetic integration fixtures, not deployments.

9 Action Types and Privileged Actions

Action type	Meaning
<code>read</code>	retrieves or inspects information
<code>write</code>	creates or modifies information
<code>external_send</code>	sends information outside the immediate system boundary
<code>delete</code>	removes information or records
<code>export</code>	packages or transfers information for downstream use
<code>purchase</code>	initiates or prepares a purchase action
<code>approve</code>	approves, confirms or authorizes a workflow step
<code>escalate</code>	escalates to a human, queue or supervisory process
<code>other</code>	action type not otherwise classified

For warning purposes the validator treats `external_send`, `write`, `delete`, `export`, `purchase` and `approve` as privileged. If these actions lack authority or review indicators, the validator emits warnings W018 and W019. The warnings do not invalidate the pack, but they surface issues for review.

10 Control Status

Tools, actions and policy controls can carry a control status of `implemented`, `partial`, `planned` or `not_applicable`. Control status is informational. In a hand-authored pack it records the author's assertion. In an imported pack it is set conservatively: the Manifest's declarations, including its review and approval declarations, are recorded as `planned`, because declaration alone does not establish implementation; independent operational verification is likewise `planned` until such evidence is supplied. Only the import's own act of running the accepted Replay semantic validator is recorded as `implemented`, and the description states that this is import-time evidence checking, not proof that runtime controls were operationally effective. Section 17.5 describes the finer-grained evidence levels that imported packs carry alongside control status.

11 Risk Register

The evidence pack includes a risk register so that governance review can connect deployment evidence to open risk. Severity values are `low`, `medium`, `high` and `critical`; status values are `open`, `mitigated`, `accepted`, `transferred` and `closed`.

The validator treats a critical open risk as incompatible with full approval: a pack with status `approved` that contains a critical open risk fails validation. This does not certify that non-critical risks are acceptable; it is a minimal consistency rule. Imported packs leave the risk register empty, and the rendered summary states that the absence of incident records does not prove the absence of incidents.

12 Validation

The validator checks evidence-pack consistency and emits a `ValidationReport`. Errors invalidate the pack. Warnings indicate review concerns but do not make it invalid.

12.1 Error Conditions

- E001. `pack_id` must be non-empty.
- E002. `title` must be non-empty.
- E003. `generated_at` must be non-empty.
- E004. `agent_overview.agent_name` must be non-empty.
- E005. `agent_overview.business_purpose` must be non-empty.
- E006. `deployment_context.environment` must be present.
- E007. Tool names must be unique.
- E008. Action names must be unique.
- E009. Every action must reference a known tool.
- E010. Blocked-action counts must not be negative.
- E011. Reliance counts must not be negative.
- E012. Risk IDs must be unique.
- E013. Replay bundle IDs must be unique.
- E014. Redaction export IDs must be unique.
- E015. Approved or approved-with-conditions status requires at least one review record.
- E016. Approving review records require a review timestamp.
- E017. A critical open risk prevents full approval.
- E018. Authority-required actions require an authority model.
- E019. Review-required actions require a policy control mentioning review or approval.
- E020. Full approval requires at least one replay bundle.

12.2 Warning Conditions

Warnings W001–W020 flag a missing owner or business unit; empty systems touched, data domains, tool inventory, action inventory, policy controls, blocked actions, reliance summary, replay bundles, redaction exports, risk register or review records; a missing validation summary; a production deployment that is still in draft or in review, has no redaction exports or has no signed bundles; privileged action types lacking review or authority indicators; and external-system tools lacking a data classification.

12.3 Validation of Imported Packs

For packs that carry `metadata.traceable_import`, validation goes further. Public validation reconstructs the material trace and re-runs the accepted Replay semantic validation over the retained records; rendering rejects inconsistent generated traces. If the required Replay validator or its compatibility mapping is not available, validation fails closed with an error (`E_TRACE_SEMANTICS`) rather than accepting the trace unchecked. In practice this means an imported pack is validated only in an environment that supplies the pinned Replay dependency, which the repository’s CI and the hub runner both do.

Validation remains implementation-level. It does not prove that evidence is complete, that controls are sufficient, that runtime enforcement exists or that compliance obligations have been satisfied.

13 Markdown Rendering

Rendering matters because governance evidence must be readable by people, not only consumed by machines. The standard renderer produces a report with a metadata table and fifteen sections: Executive Summary; Agent Overview; Deployment Context; Tool Inventory; Action Inventory; Authority Model; Policy Controls; Blocked-Action Summary; Reliance Summary; Replay Bundle Inventory; Validation Summary; Redaction and Export Summary; Risk Register; Review Records; and Known Limitations.

The executive summary includes counts of tools, actions, privileged actions, bundles, validation results and open risks. Output is deterministic, makes no external calls, escapes table delimiters and renders empty sections with an explicit “No X recorded” placeholder.

The traceable renderer, invoked from the command line as `agep render --traceable`, appends a sixteenth section, *Traceability and Import Provenance*, generated from `metadata.traceable_import`. It contains readable tables for input artifacts, transformation, derived counts and counting rules, lifecycle and effect status, control evidence levels, import findings, unresolved issues, producer profiles, the executor producer contract, conversion losses, source-record references and source-supplied and locally computed commitments, followed by the complete material trace as escaped JSON. The purpose is that a reviewer who reads only the Markdown still has the equivalent material information.

The rendered Markdown is not a separate source of truth. It is a deterministic presentation of the underlying pack.

14 Command-Line Interface

The `agep` command provides small utilities for local review and automation.

```
agep validate <evidence_pack.json>
agep summarize <evidence_pack.json>
agep render <evidence_pack.json> [--out <output.md>] [--traceable]
agep import --manifest <manifest.json> \
    --reconstruction <bundle.json> \
    --out <evidence_pack.json> [--render <output.md>]
agep check-examples
```

`validate` prints errors and warnings and exits 0 when valid, 1 when invalid and 2 on usage, file, parse or model errors. `summarize` prints compact counts and status fields, and for imported packs adds the distinct effect count, the separate Control Plane and executor attempt counts and the latest destination observation. `render` produces the standard or traceable report. `import` builds a traceable pack from a Manifest and a Reconstruction Bundle and can render it in the same step. `check-examples` validates every top-level example pack and regenerates its rendered report.

These commands are intended for local validation, examples and lightweight workflows. They are not a hosted service, daemon, dashboard or enterprise control surface.

15 JSON Schemas and Public Conformance

The repository includes JSON Schema files for the public evidence-pack objects; the primary schema is `evidence_pack.schema.json`, with one schema per field group and for the validation issue and report. The schemas serve interchange, structural validation outside Python, documentation of the public boundary and a basis for future conformance tests.

The schema boundary is deliberately limited. Schemas validate structure, required fields, types and allowed values. They do not prove that a pack is operationally sufficient or that an obligation has been satisfied. The `traceable_import` provenance object is carried in the free-form `metadata` field; its consistency is checked by the validator and the accepted Replay validator, not by the JSON schema. The imported pack format is versioned as schema 0.2.0, and that version has not changed across the transformation versions described below.

16 Example Evidence Packs

The repository includes two kinds of examples.

16.1 Hand-Authored Scenario Packs

Four hand-authored packs illustrate common enterprise agents: a **customer-service agent** that reads CRM records and whose outbound email sends were blocked for missing authority; an **internal research agent** with document access, export controls and open review concerns; a **procurement agent** with purchase authority controls and approval conditions; and a **finance workflow agent** with privileged-action controls, approval requirements and audit exports. All four validate; each except the procurement pack carries warnings that make its remaining gaps visible (the customer-service pack, for example, is still `in_review` with no review record). They are illustrations of the format, not evidence of any deployment.

16.2 Imported Synthetic-Refund Packs

A second set of examples was generated by the importer from actual producer-profile 2.0.0 runs of the synthetic refund workflow: a **lost acknowledgement**, a **partial delivery**, a **restart** recovery, a **prior observed absence** and a **rejected wrong-effect observation** followed by later recovery. Each is retained as a JSON pack and a traceable Markdown rendering. In the lost-acknowledgement case, for example, the pack records one proposal, one decision, one distinct effect and one attempt in each of the Control Plane and executor namespaces; authorization granted but labeled historical, with current permission not evaluated; acknowledgement unknown and destination observed applied, both retained; reconstruction complete; retry permission not established; and independent verification unavailable. These packs were produced under transformation 0.3.0 and remain attributed to it.

17 Traceable Imports

The first edition listed “optional import helpers from replay bundles” and “from action manifests” as near-term extensions. Both now exist as a single deterministic import contract.

17.1 Inputs and Transformations

The importer consumes one Action Manifest 1.1 artifact and one Reconstruction Bundle 0.2.0 artifact. It does not fetch external references and treats source text as data. It accepts only producer revisions it explicitly supports and fails closed on unsupported or contradictory revision and profile attribution. Each supported producer generation is mapped by its own versioned transformation, and historical outputs are never relabeled:

Transformation	Producer generation
agep-manifest-reconstruction-import/0.2.6	historical executor producer profile 1.0.0 and legacy unversioned executor artifacts
.../0.3.0	executor producer profile 2.0.0 with the earlier Control Plane v2 revision
.../0.3.1	the persistence-repaired Control Plane generation
.../0.3.2	the exact merged Control Plane and executor pair selected by the hub

The 0.3.2 mapping covers the bounded workflow export 2.0.0 for exact revisions of the Action Manifest, Replay Bundle, Control Plane and Execution Runtime. It does not add atomic-claim or refund-intent evidence. The component's own checkpoint left ODES and GAX composition with this mapping to a separate integration gate; that composition was qualified at the hub level, as part of the merged consumer chain the lock selects.

17.2 Provenance Contract

Every imported pack records, under `metadata.traceable_import`, the transformation version; the canonicalization profile (sorted-key compact UTF-8 JSON without NaN); the identity, version and SHA-256 digest of each input artifact; the producer revisions selected for this import separately from the revision sets the importer supports; derived counts with explicit coverage denominators; the lifecycle summary; the control evidence levels; source-record references; source-supplied commitments alongside locally computed ones; import findings and a conversion-loss register; any manual assessments; and a list of unresolved issues. Imported facts, computed summaries and manual assessments remain separate, and manual additions require attributable provenance and do not overwrite generated findings.

17.3 Counting Rules

The importer counts carefully because counts are what executives read first. Proposals are counted by distinct action or proposal identifier, decisions by distinct `decision_id` and effects by distinct `effect_id`, so duplicate submissions and restart records do not inflate them. Control Plane and executor attempts are separate namespaces unless explicit correlation exists, and repeated lifecycle transition records do not inflate attempt counts. Missing observations are represented as unavailable; they are not counted as zero, failed or successful effects.

17.4 Lifecycle Distinctions

The lifecycle summary keeps separate: authorization granted or held; current permission, which historical records cannot establish; execution attempted; acknowledgement received, unknown or not applicable; destination observed as absent, applied, partial, unknown or unavailable; reconstruction completeness; retry permission; and independent verification. These separations follow the stack's recovery semantics. A lost acknowledgement followed by an applied observation retains both facts. A historical rejected observation may coexist with a later accepted applied observation. An accepted `observed_absent` reconciliation with `retry_eligible: false` remains explicit and does not create retry permission. Reconstruction completeness does not mean effect completion, and an observed local destination state does not establish an independently verified institutional outcome.

17.5 Evidence Levels

Control evidence is graded on separate levels, each with its own status and reference:

Level	Meaning and default
declared	a Manifest or producer record says a control or requirement exists
implemented	requires implementation-specific evidence; not inferred from declarations or import success
semantic validation during import	the accepted Replay validator ran over the retained records; not test-run evidence
source-asserted runtime evidence	producer metadata asserts fixture or runtime provenance; does not prove a test occurred
attributable test-run evidence	complete source-attributed provenance with run identity, producer, scope and result; malformed provenance remains unavailable and produces a finding
tested	unavailable unless valid attributable test-run evidence is supplied, and then only for its stated scope; source outcomes such as failed or inconclusive are preserved
operationally observed	requires real deployment observation; unavailable unless supplied
independently audited	requires independent evidence; unavailable unless supplied

None of these levels supports a statement that a control is operationally effective, that a deployment is approved or that it has been independently audited unless that specific evidence was supplied. The pack also separately records that the import did not execute the Evidence Pack repository's own test suite.

17.6 Integrity and Redaction Limits

Where an upstream producer uses HMAC, it establishes shared-secret integrity under that producer's profile; it does not establish public issuer identity or independent review. Producer evidence classes and commitment verification statuses remain source assertions. Original and redacted derivative identities remain distinct, and original commitments are never presented as verification of modified redacted content. If a source does not declare its redaction state, the importer records a finding that the pack's boolean redacted flag defaults to false only for schema compatibility, so the gap stays visible.

17.7 Known Limits of the Import Path

The import path consumes one proposal and envelope operation at a time and does not imply fleet-wide or multi-operation coverage. Known producer gaps, such as Control Plane institution and domain fields and the distinction between Authority Context profile references and context-instance identifiers, are surfaced as findings rather than repaired. Production resolver authentication, host confinement, distributed budgets and independent real-world effect verification are outside every supported producer baseline.

17.8 Supplementary Projections

The repository also contains narrower transformations outside the selected release. A merged, optional *recovery inspection* transformation (`agep-recovery-inspection/0.1`) turns a Replay recovery-inspection artifact into a native, read-only evidence artifact. It retains the complete Replay input, re-runs

the actual Replay validator, records review status as `unreviewed`, sets settlement verification and retry eligibility to false and execution authority to none, and fails rather than falling back to a permissive parser. Separate lineage projections preserve atomic refusal history and retained stop records without promoting them to authority. These are qualified by their own workflows against pinned candidate revisions; they are not part of the hub's selected lock, and they are not instances of the `EvidencePack` model.

18 Relationship to Runtime Evidence

Governance Evidence Pack is not a runtime recording layer. It depends on evidence produced elsewhere:

- the Action Manifest supplies declared tools, actions and requirements;
- Control Plane records supply authorization decisions, holds, denials, effect-time revalidation outcomes and reconciliation;
- Execution Runtime records supply executor attempts and acknowledgements under a versioned producer profile;
- the Replay Bundle reconstructs those records into a validated Reconstruction Bundle with findings and conversion losses;
- governance workflows supply risk and review records;
- redaction and signing processes supply export and integrity metadata.

The pack does not require a specific upstream implementation for hand-authored use. The traceable import, however, is bound to exact supported producer revisions, because a mapping that silently accepted any producer would undermine the provenance it records.

Two properties of the upstream stack matter for interpreting any pack. First, the selected Control Plane mode (C0) revalidates decision-critical inputs immediately before dispatch; this is pre-effect revalidation, not atomicity with the destination commit, and a check-to-commit race remains. An optional same-host SQLite authority/effect profile (C1) exists but is not enabled by default, and no C2/C3 claim is made. Second, effect-ID deduplication is not business-intent deduplication. An optional refund-intent profile exists in the selected runtime source, but the default path does not enable it. An evidence pack faithfully summarizing a run under these conditions inherits their limits; it does not remove them.

19 Security Considerations

Governance Evidence Pack improves reviewability and governance transparency, but it should not be treated as a security product. Important limitations include:

- the pack summarizes records but does not enforce policy or grant authority;
- no pack, digest or signature authorizes execution, renews permission or establishes retry eligibility;
- review status is not compliance certification;
- validation checks structure and consistency, not operational sufficiency;
- blocked-action and reliance summaries depend on upstream records and provenance capture;
- imported provenance, producer evidence classes and commitment statuses are source assertions, not independent provenance verification;

- locally computed commitments check content consistency, not source authenticity; coordinated fabrication of unauthenticated source and derived data is outside the consistency guarantee;
- redaction summaries do not guarantee that all sensitive data was removed;
- signed bundle indicators and shared-secret integrity do not replace production key management or public issuer identity;
- risk registers are only as complete as the review process that maintains them;
- production deployments still require authentication, authorization, monitoring, incident response, runtime control and organizational governance.

The correct security posture is layered. The evidence pack supplies structured, traceable business-facing evidence; it must be combined with independently supplied authority, runtime control, constrained execution, reconstruction, monitoring, application authorization, risk management and governance review.

20 Implementation Boundary

The public reference implementation is intentionally thin. It provides enough structure to demonstrate the evidence-pack category and support real integration experiments, and it does not depend on private Cog-nous product internals. Consistent with the stack’s open-source boundary, the public layer opens formats, schemas, validators, reference behavior, examples, tests and business documentation; a commercial layer may add hosted services, integrations and operations.

Public reference layer	Possible product layer
Evidence pack schema and object model	Hosted evidence-pack store
Semantic and trace validation	Automated evidence completeness scoring
Standard and traceable Markdown renderers	Hosted review interface and dashboards
Traceable importer for supported producers	Automated ingestion from customer systems
Inventory, authority and policy summaries	Policy authoring and enterprise action registry integration
Replay bundle inventory and lifecycle facts	Replay viewer and investigation tooling
Redaction export summary	Production export management
Risk register and review records	Enterprise risk, identity and approval workflow integration
CLI utilities and examples	Platform APIs, managed services and deployment templates
JSON schemas	Compliance mappings and enterprise conformance tooling

Public software should expose the integration contract: what evidence can be summarized, how it is traced, what validators check and what downstream systems can render. Product systems can add enterprise integrations, hosted workflows, identity management, automated evidence ingestion, dashboarding and customer-specific governance logic. The right-hand column describes possible extensions, not shipped products.

21 Reference Implementation Status

Capability	Status	Notes
Core object model	Implemented	Pydantic v2 models; Python 3.11+
Loader and dumper	Implemented	JSON loading, model validation, formatted export
Validation reports	Implemented	E001–E020 errors, W001–W020 warnings
Inventories, authority, controls	Implemented	tools, actions, scopes, control status
Blocked actions and reliance	Implemented	counts, reasons, sources, references
Replay bundle inventory	Implemented	signed, redacted, validation status
Validation, redaction, risk, review	Implemented	summaries, recipients, register, decisions
Markdown renderer	Implemented	fifteen-section business report
Traceable importer	Implemented	Manifest 1.1 + Reconstruction Bundle 0.2.0; transformations 0.2.6–0.3.2
Assurance separation	Implemented	lifecycle facts, evidence levels, findings, losses
Trace validation	Implemented	re-runs accepted Replay validator; fails closed
Traceable renderer	Implemented	sixteenth section plus escaped trace
CLI	Implemented	validate, summarize, render, import, check-examples
JSON schemas	Implemented	public schema boundary
Example packs	Implemented	four hand-authored; five imported synthetic-refund
Test and qualification suites	Implemented	unit, producer, persistence and merged-pair suites
Recovery inspection evidence	Merged, optional	qualified separately; not in selected lock
Tenant- and profile-aware lineage	Contract defined	future mapping requirements (W0 C8 addendum)
Hosted review UI	Extension	out of current scope
Automated enterprise ingestion	Extension	out of current scope
Enterprise approval workflows	Extension	out of current scope
Compliance mapping	Extension	out of current scope

22 Test Suite, Qualification and Stack Evidence

Tests are not part of the evidence-pack format, but they are part of its public trust surface: they show which claims are exercised by executable checks.

22.1 Component Tests

The repository's test suites cover model, loader, validator, renderer, summary, CLI and example behavior, including duplicate identifiers, unknown tool references, approval without review records, critical open risk, authority- and review-required actions without supporting controls, production and privileged-action warnings, and CLI usage errors. Importer and producer tests cover counting rules, lifecycle distinctions, revision and profile attribution, rejection of ambiguous or dangling attempt lineage, operation and attempt lineage tampering, malformed or missing test provenance, failed source-attributed test results, and the separation of Control Plane and executor namespaces.

22.2 Qualification Workflows

GitHub Actions runs the main suite on Python 3.11 and 3.12 against checked-out, revision-verified producer repositories, imports an actual pinned producer example and then validates and renders it. The main workflow also qualifies the persistence-repaired Control Plane path, and a dedicated workflow qualifies the exact merged producer pair. In that workflow, the actual producer workflow runs 15 lifecycle scenarios, and import, validation and rendering occur inside assertions that the retained Control Plane record bytes and the logical SQLite destination contents remain unchanged. This establishes that the exercised Evidence Pack path is non-effecting with respect to those bounded synthetic stores; it does not establish production confinement. Missing configured checkouts fail the qualification rather than being skipped.

22.3 Stack-Level Evidence

The hub's component lock (profile `merged-producers-v1`) selects the Evidence Pack revision carrying transformation 0.3.2. The current release evidence for the stack resolved all 35 acceptance scenarios, with required suites executed twice (CI run 37694032916). A fresh reproduction on 13 October 2026 executed 1,009 Python testcases in each of two repetitions plus 20 Node tests, with 34 invariants passing and one characterized limitation. These are aggregate stack results, not a per-component count. The optional C1 profile was qualified separately (73 tests) and is not added to the release count. The 915-test figure cited in earlier collateral belongs to a historical evidence generation and is not current.

None of this establishes production readiness. The stack is a merged, pinned, bounded synthetic reference on one host. Feature intake was frozen for release closeout on 13 October 2026; public-preview approval is pending, and final publication and operational trust remain on hold.

23 Example Use Cases

23.1 Deployment Review

An AI platform team prepares to move an agent toward production. The evidence pack summarizes purpose, systems touched, tools, actions, controls, blocked actions, reliance records, validation results, exports, open risks and review decisions, so the review board receives one structured package rather than scattered artifacts.

23.2 Security Review

A security organization reviews whether an agent has appropriate controls. The pack shows privileged action types, authority and review requirements, policy controls, blocked behavior, data classifications, redacted exports and signed bundle availability, and the evidence levels show which controls are only declared. Security can identify missing controls or approve with conditions.

23.3 Audit Preparation

An audit team needs evidence that a deployed agent was governed. The pack summarizes the available records, reconstruction bundles, validation status, exports and review decisions, with input digests and producer revisions. It does not replace the underlying records; it provides the index and narrative auditors need to inspect them.

23.4 Incident Review

A refund request times out and nobody knows whether it was applied. The imported pack identifies the bundle, shows the unknown acknowledgement and the later destination observation as separate facts, states that retry permission is not established and that historical authorization does not imply current permission, and points to the exact source records. Investigators move from business summary to technical evidence without rebuilding the trail or mistaking uncertainty for failure.

23.5 Executive Reporting

Executives need a concise view of deployment posture. The rendered pack provides purpose, context, counts of tools and actions, privileged-action count, bundle availability, validation results, open risks, review decision and, for imported packs, what remains unresolved.

24 Evaluation Criteria

The usefulness of Governance Evidence Pack should be evaluated against operational criteria rather than benchmark scores. A deployment should ask:

1. Does the pack identify the agent, its business purpose, owner and business unit?
2. Does it describe the deployment environment, systems touched and data domains?
3. Does it list tools and actions with their types and control status?
4. Does it identify authority-, review- and reliance-required actions?
5. Does it summarize authority models and policy controls?
6. Does it summarize blocked actions and reliance sources?
7. Does it inventory reconstruction bundles or run records, and summarize validation results?
8. Does it identify redacted or signed exports, keeping derivatives distinct from originals?
9. Does it include known risks with severity, status, mitigation and owner, and review records with decisions?
10. Does it validate with no error-severity issues, and are its warnings useful for review?
11. Can it be rendered into a business-facing report without losing the material trace?
12. Can reviewers trace from each summary claim to an evidence reference and source record?
13. Which exact input, output and producer revisions does it rely on?
14. Does it distinguish declared, implemented, tested, observed and independently audited evidence?

15. Does it keep acknowledgement, destination observation, reconstruction completeness, retry permission and current permission separate?
16. Are missing, unknown, unavailable and redacted states preserved rather than normalized?
17. Which documented checks were actually executed in the intended environment, and what deployment-specific work remains before relying on the result?

25 Roadmap

The public roadmap should remain implementation-focused and avoid overclaiming. Since the first edition, the traceable importer, assurance separation, trace validation and traceable rendering have been implemented, satisfying the earlier “import helper” items. Remaining candidate extensions recorded in the repository include evidence-pack diffing between review cycles, richer executive summaries, CSV and DOCX export, additional industry examples and a formal conformance suite for third-party implementations.

The W0 C8 compatibility addendum defines requirements for a future tenant- and profile-aware transformation: it must preserve exact producer and Replay revisions; tenant lineage and explicit missing or unsupported tenant state; refusal and failure classification; optional profile identity, configuration digest and capability admission; flow-decision and result-admission outcomes; distinctions among stop request, acknowledgement, dispatch closure, quiescence and reconciliation; and observation state, freshness, finality and uncertainty. A missing lineage field cannot be converted into successful exact-scope evidence. This is a contract, not a selected interface.

Release closeout froze feature intake on 13 October 2026, so none of these extensions is part of the forthcoming bounded preview. Longer-term work such as hosted review interfaces, compliance mapping, approval workflows, automated ingestion, dashboarding and governance-system integrations are product-layer concerns and should remain separate from the minimal public reference.

26 Conclusion

AI-agent governance cannot end with technical runtime records. As agents gain access to tools, systems, data and workflows, enterprises need business-facing evidence packages that summarize what the agent is for, where it is deployed, what it can do, what controls apply, what was blocked, what sources were relied on, what reconstruction records exist, what validation found, what remains unresolved, what exports were redacted or signed, what risks remain open and who reviewed the evidence.

Cognous Governance Evidence Pack provides a public reference for that evidence work within the Replay & Evidence group of the Cognous Open Control Stack. It defines the object model and schemas, validates consistency, renders business-facing reports and, for supported producers, imports Manifest and Reconstruction Bundle artifacts into packs that record their own provenance, keep lifecycle facts apart and grade the strength of their evidence.

The contribution remains deliberately narrow. Governance Evidence Pack does not run agents, enforce policy, grant authority, verify settlement, certify compliance or prove correctness. Its selected revision has been exercised only within a bounded synthetic reference. It supplies a concrete, traceable evidence artifact for a specific operational need: making AI-agent governance review inspectable, structured and portable, without allowing a summary to claim more than its sources support.

References

- Cognous. *Cognous Governance Evidence Pack*. Public reference implementation, 2026.
<https://github.com/cogno-us/cognous-governance-evidence-pack>. Documentation on format, traceable imports, validation, rendering, merged-producer compatibility and recovery inspection.
- Cognous. *Cognous Open Control Stack*: hub, component lock and release status. 2026.
<https://github.com/cogno-us/cognous-open-control-stack>
- Cognous. *Cognous Action Manifest*. 2026.
<https://github.com/cogno-us/cognous-action-manifest>
- Cognous. *Cognous Control Plane*. 2026.
<https://github.com/cogno-us/cognous-control-plane>
- Cognous. *Cognous Replay Bundle*. 2026.
<https://github.com/cogno-us/cognous-replay-bundle>
- Cognous. *Open Decision Evidence Standard (ODES)*. 2026.
<https://github.com/cogno-us/open-decision-evidence-standard>
- National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1, 2023.
- OWASP Foundation. *OWASP Top 10 for Large Language Model Applications*. Project documentation.
- OECD. *Agentic AI in organisations: Early insights from practitioner interviews*. OECD Artificial Intelligence Papers, No. 65, 2026.
- Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.