

Cognous Replay Bundle: Portable Reconstruction Records for AI-Agent Runs

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

October 2026

Abstract

AI agents increasingly operate through tools, APIs, data sources, workflow systems and enterprise applications. When those agents begin to propose consequential effects, such as refunds, messages or record changes, organizations need more than logs. They need portable records that let an agent run be reconstructed, validated, redacted, signed, reviewed and attached to downstream governance evidence. Those records must also stay honest about what remains unresolved. Cognous Replay Bundle (formerly Cognous Agent Replay Bundle) is a public reference format, importer, validator and command-line tool for this purpose. It retains the original replay-bundle format for run frames, action proposals, policy decisions, evaluation traces, blocked actions, authority and reliance records, validation, redaction and signature metadata. It adds Reconstruction Bundle 0.2.0, a separate model that imports the records actually retained by exact, revision-pinned producer versions of the Cognous Control Plane and Execution Runtime. The importer preserves proposal, decision and effect identity. It keeps the Control Plane and executor attempt namespaces apart, retains unknown acknowledgements and rejected observations, and reports missing or unsupported information explicitly. Reconstruction is non-effecting: it does not rerun a model, re-evaluate policy, renew authorization, repeat an effect or independently verify delivery. The component is selected, through its Reconstruction Bundle 0.2.0 interface, in the Cognous Open Control Stack's merged and pinned bounded synthetic reference. It is not a production deployment, a compliance certification or evidence of institutional adoption. Its contribution is narrow and operational: a structured reconstruction package that helps teams preserve the technical evidence needed to establish what an agent proposed, what was decided, what was attempted, what was observed, and what remains unknown.

1 Introduction

AI-agent systems are moving from conversational support toward delegated action. Agents can retrieve data, call tools, draft communications, prepare exports and propose operations that change records in other systems. As these systems touch more business processes, organizations need a way to reconstruct agent behavior after the fact.

Traditional application logs are useful, but they are rarely sufficient for agent governance. A log line may show that a function was called. It may not show what the agent proposed before execution, what policy decided, what was blocked, what authority was in force, what sources the agent relied on, whether an attempted effect was acknowledged, what the destination was later observed to contain, or whether an exported record can be checked for modification.

Replay Bundle addresses this reconstruction problem. A replay bundle is not the runtime. It is a record package produced by or from runtime systems, and its purpose is to make a run inspectable after execution. Since the July 2026 version of this paper, the component has gained a second, stricter role. Besides packaging run evidence in its original format, it now imports the actual retained records of specific Cognous producers. In doing so it preserves identity, lineage and uncertainty without converting any of them into claims the sources do not support.

The guiding design principle is:

No enterprise agent run should be considered reviewable unless the organization can reconstruct what the agent proposed, what was decided, what was attempted, what was observed, which source supplied each fact, and what remains unknown.

The public reference implementation exposes two object families:

- the original family (format version 0.1): `AgentReplayBundle`, `RunFrame`, `ActionProposal`, `PolicyDecision`, `PolicyEvaluationTrace`, `PolicyRuleEvaluation`, `BlockedAction`, `AuthorityRecord`, `RelianceRecord`, `ValidationIssue`, `ValidationReport`, `RedactionMetadata`, `SignatureMetadata` and `SignedReplayBundle`;
- the reconstruction family (Reconstruction Bundle 0.2.0): `ReconstructionBundle`, `ProducerProfile`, `SourceRecord`, `RecordLink`, `CommitmentRecord`, `ImportReport`, `ImportFinding`, `ReconstructionSemantics`, `DerivationMetadata` and `IntegrityMetadata`.

The repository also provides revision-pinned importers, a command-line interface, JSON schemas, example bundles, validation, redaction and signing helpers, and test suites.

A terminological note: in this paper, “replay” means reconstruction of retained records, never re-execution. The Open Control Stack hub separately uses “paired replay” for an enforcement experiment that compares an identical frozen request across enforcement conditions. That experiment is not Replay Bundle reconstruction and is not discussed further here.

2 Problem Statement

Enterprise AI-agent governance has a replay gap. Runtime behavior may occur quickly, across multiple tools and systems, under changing policy and authority conditions. Records from authority evaluation, execution and recovery can disagree or omit important facts. After the fact, reviewers need a portable package that can answer a precise question:

What happened during this agent run, which records support each fact, and what evidence exists to review it?

This gap appears in several operational forms.

2.1 Logs Are Not Replay Packages

Logs may record low-level events, errors or service calls. They do not necessarily preserve agent-level proposals, decisions, attempts, observations and their sources in one coherent package. A replay bundle is organized around the run. It collects the evidence needed to reconstruct and inspect the run rather than leaving reviewers to infer behavior from scattered logs.

2.2 Action Proposals Are Often Lost

For governed agents, the proposal matters. An agent may propose an action that is never executed because policy blocks or holds it. If only executed actions are logged, reviewers lose evidence of what the agent attempted to do. Replay bundles preserve proposals regardless of outcome.

2.3 Blocked Actions Need First-Class Records

Blocked actions are not failures to be hidden. They are governance evidence. A blocked outbound email, a blocked purchase order or a denied refund may show that controls operated as intended. The original format records blocked actions explicitly. The reconstruction importer retains blocked and held decisions together with the producer's stated reasons.

2.4 Policy Decisions Need Context

A policy decision is not just an outcome. Reviewers need to know which proposal it applied to, what result was returned, which policy, grant and grant revision were in force, why the decision was made, and when it occurred. In the bounded workflow, a decision is also bound to an exact proposal commitment. Reviewers need to see that binding, not just a label.

2.5 Policy Evaluation Needs Traceability

A decision such as block or allow is easier to review when the rules leading to it are preserved. The original format's policy evaluation trace records individual rule evaluations, results, explanations, the final result and a deterministic fingerprint, so reviewers can inspect the decision path as well as the outcome.

2.6 Authority and Reliance Are Often Reconstructed Too Late

After an incident or audit, teams may ask whether the agent had authority and what sources it relied on. If those records were not captured during or near the run, reconstruction becomes costly and unreliable. The original format carries authority and reliance records. The reconstruction importer separately retains authority-context identifiers, grant and policy versions as the producers supplied them.

2.7 Outcomes Can Be Uncertain

A consequential effect does not always end in a clean success or failure. A caller can time out before acknowledgement while the effect commits anyway. A destination observation can be stale, malformed or tied to the wrong effect. A later reconciliation can resolve what an earlier attempt left unknown. A reconstruction that collapses these states into a single status tells a retrospective success story. It also risks implying that an unresolved effect may be retried. Reviewers need unknown, rejected and missing evidence to remain visible as such.

2.8 Exports Need Redaction and Integrity Metadata

Run records may contain sensitive payloads, targets, reasons or outputs, and they may need to be shared with internal reviewers, auditors, customers, counsel or regulators. Those exports need redaction metadata and integrity metadata. They also need rules that prevent a stale signature or digest from appearing to cover modified content.

3 Design Goals

Replay Bundle is designed around twelve goals. The first ten carry over from the original design; the last two were added with Reconstruction Bundle 0.2.0. The 0.2.0 work also strengthened goals 8 to 10.

1. **Run-centered packaging.** Records should be organized around a specific agent run.
2. **Proposal preservation.** Proposals should be preserved whether allowed, blocked, held or escalated.
3. **Decision capture.** Policy decisions should be explicit, inspectable and linked to the proposals they govern.
4. **Traceability.** Rule-level evaluation detail should be preserved where available.
5. **Blocked-action visibility.** Blocked actions should be first-class records.
6. **Authority visibility.** Authority context should be associated with the run as supplied, not backfilled.
7. **Reliance visibility.** Source reliance should be captured and linked to actions where possible.
8. **Semantic validation.** Bundles should be checked for internal consistency beyond schema validation. Imported producer records should be checked as one cross-record chain.
9. **Safe export preparation.** Bundles should support redaction that preserves governance-relevant structure and gives derivatives their own identity.
10. **Export integrity.** Bundles should support shared-secret integrity metadata without claiming production key management.
11. **Exact producer compatibility.** Imports should accept only producer revisions and profiles they were qualified against. Similar shape is never treated as version negotiation.
12. **Non-effecting, loss-explicit reconstruction.** Import should never re-evaluate policy, renew authority or create effects. It should report missing, unknown, unavailable and unsupported information instead of hiding it.

4 Non-Goals

The architecture is deliberately bounded. Replay Bundle is not:

- an agent framework, model runtime or planner;
- a policy engine or runtime control plane;
- a mechanism for re-evaluating policy or renewing authorization;
- an executor, or a way to repeat or retry an effect;
- an independent verifier of destination state, delivery or settlement;
- a replay viewer or hosted service;
- a complete enterprise governance platform;
- a production authorization system or a security boundary by itself;
- a substitute for application-level authorization;
- a guarantee that model outputs are true, safe or complete;

- a guarantee that regulatory obligations have been satisfied.

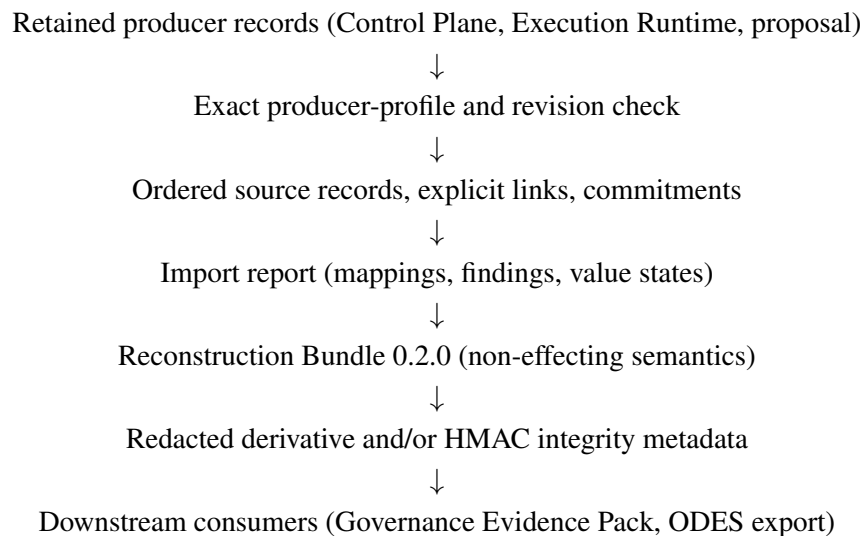
It packages and imports run evidence and validates its internal consistency. It does not execute actions, enforce policy, govern tool access, certify compliance or prove that a deployment is safe. A valid signature, chain inclusion, message receipt, reasoning instruction or evidence-package digest never authorizes execution.

5 System Overview

Replay Bundle now supports two complementary paths.

Packaging path (format 0.1). Any system can package a run frame, action proposals, policy decisions, evaluation traces, blocked actions, authority and reliance records, and a final output into a 0.1 replay bundle (`AgentReplayBundle` object). The bundle can then be validated, summarized, redacted, signed, stored or shared. This path is framework-neutral and does not require any Cognous producer.

Reconstruction path (Reconstruction Bundle 0.2.0). A revision-pinned importer consumes records actually retained by a supported producer. The input can be a legacy Control Plane replay export, a Control Plane bounded run record, the original runtime proposal, or an Execution Runtime producer export. The importer emits a `ReconstructionBundle`:



The 0.1 format is not silently reinterpreted. A legacy `status="complete"` must not be mapped to effect completion, and 0.2.0 bundles do not reuse 0.1 field semantics. This separation is intentional. Forcing decision bindings, effects, attempt lifecycles, observations and reconciliations into the 0.1 fields would either lose producer semantics or invent fields that never existed.

5.1 Position in the Cognous Open Control Stack

The July paper placed Replay as the third layer of a four-step sequence: Declare, Control, Replay, Evidence. The current stack is organized differently. It has thirteen components in six responsibility groups: 01 Knowledge & Provenance; 02 Institutional Governance; 03 Declaration & Policy Control; 04 Constrained Execution; 05 Replay & Evidence; and 06 Reasoning & Human Oversight. Governed Exchange (GAX/IMX) is a cross-cutting integration layer. The groups are a responsibility map, not an execution sequence.

Replay Bundle belongs to group 05 with the Governance Evidence Pack and ODES. In the single supported workflow, a synthetic refund, the sequence runs as follows. A governed message is delivered through local durable transport and assessed by the GAX recipient adapter. The adapter produces a proposal bound to an Action Manifest (Manifest 1.1 runtime bindings). A separately configured resolver supplies Authority Context. The Control Plane decides and revalidates decision-critical inputs at effect time, and keeps persistent records of decisions, attempts and reconciliations. The Execution Runtime reaches a synthetic SQLite destination. Retained Replay, ODES and IMX artifacts then feed a Governance Evidence Pack. Replay's responsibility in that chain is non-effecting reconstruction from retained producer records, preserving identities and provenance. Policy re-evaluation, renewed authority and independent verification are explicitly outside it.

6 Core Object Model

6.1 Format 0.1: AgentReplayBundle

An `AgentReplayBundle` is the top-level object of the original format. It is unchanged since July 2026 and remains available for compatibility.

Field	Meaning
<code>bundle_id</code>	unique replay bundle identifier
<code>bundle_version</code>	replay bundle format version (default 0.1)
<code>run_id</code>	identifier of the agent run
<code>status</code>	complete, partial or redacted
<code>generated_at</code>	timestamp when the bundle was generated
<code>producer</code>	optional system that produced the bundle
<code>frame</code>	run frame describing the run context
<code>action_proposals</code>	proposed actions made by the agent
<code>policy_decisions</code>	decisions made in response to proposals
<code>policy_traces</code>	detailed policy evaluation traces
<code>blocked_actions</code>	actions blocked by policy
<code>authority_records</code>	authority grants in scope for the run
<code>reliance_records</code>	sources the agent relied on
<code>final_output</code>	final output produced by the run, if available
<code>validation_report</code>	optional embedded validation report
<code>redaction_metadata</code>	redaction status and redacted field paths
<code>signature_metadata</code>	optional embedded signature metadata
<code>metadata</code>	optional extension metadata

The component objects are as follows.

RunFrame records the frame ID, task, actor, environment, allowed and blocked tools, policy version and creation time. It anchors the run.

ActionProposal records the action ID, run ID, tool name, action type, target, payload, proposal time and optional reason. Proposals are preserved even when never executed.

PolicyDecision records the decision ID, action ID, run ID, result (`allow`, `block` or `escalate`), policy name, reason, decision time, deterministic fingerprint and optional trace ID.

PolicyEvaluationTrace records the trace ID, run ID, action ID, evaluation time, list of rule evaluations, final result and deterministic fingerprint.

PolicyRuleEvaluation records one rule's name, whether it matched, its result (`allow`, `block`, `escalate` or `none`) and a reason.

BlockedAction records the blocked ID, action ID, run ID, reason, policy name and time. It must correspond to a `block` decision for the same action.

AuthorityRecord records the authority ID, run ID, actor, scope, source, optional expiry and creation time. It preserves asserted authority context; it does not grant authority.

RelianceRecord records the reliance ID, run ID, source name, source type (`tool`, `database`, `file`, `api`, `user_input`, `model_output` or `other`), scope, optional referenced action and creation time.

ValidationIssue and ValidationReport record errors and warnings found during validation. A report is valid only when no error-severity issue is present.

RedactionMetadata records whether redaction was applied, when, which fields, under which policy and with which replacement value.

SignatureMetadata and SignedReplayBundle record export-integrity signature state. The signed wrapper is the authoritative signed-export object. Embedded signature metadata on the bundle is optional and excluded from the signing payload.

6.2 Reconstruction Bundle 0.2.0

A `ReconstructionBundle` is a separate model with a fixed `bundle_version` of `0.2.0`. It is designed to hold producer records without renaming them and without claiming more than they establish.

Field	Meaning
<code>bundle_id</code>	reconstruction identity (a redacted derivative receives a new one)
<code>bundle_version</code>	fixed at <code>0.2.0</code>
<code>run_id</code>	run identifier, where supplied
<code>status</code>	<code>reconstruction_complete</code> , <code>reconstruction_partial</code> , <code>incomplete</code> or <code>redacted</code>
<code>producer_profiles</code>	exact producer, repository, revision and format for each input
<code>records</code>	ordered source records with retained producer content
<code>links</code>	explicit or shared-reference links between records
<code>commitments</code>	source commitments with algorithm, canonicalization and verification status
<code>import_reports</code>	per-adapter mappings and findings
<code>semantics</code>	declared non-effecting import semantics
<code>derivation</code>	redacted-derivative lineage, when applicable
<code>integrity</code>	content digest and/or HMAC metadata
<code>metadata</code>	extension data, including effect observation history

ProducerProfile identifies one input: profile ID, producer, repository, exact revision, format name and version. Repository strings are retained wire identifiers. They keep their pre-rewrite spelling where producers emitted it.

SourceRecord holds one retained record: its type, producer profile, source sequence and path, extracted identifiers, and the producer's original data, preserved including JSON null. Each record carries an evidence class. The default is `producer_reported`. The importer never manufactures the `locally_generated` or `independently_checked` labels.

RecordLink connects two records as `explicit` or `shared_reference`, with a stated basis. Its `establishes_identity_equivalence` flag defaults to false: a shared decision or effect identifier can support lineage without making two attempts the same attempt.

CommitmentRecord retains a source commitment's value, algorithm and canonicalization profile. Its verification status records whether the importer left it unchecked, recomputed it (a checked match or mismatch), or retained it only as an attributed claim of the producer.

ImportReport and ImportFinding record explicit field mappings and findings. A finding falls into one of six categories: unmapped field, missing dependency, unsupported semantic, redaction, conversion warning or value state. Value states are `absent`, `unknown`, `unavailable` and `redacted`.

ReconstructionSemantics declares what the import did and did not do. Record reconstruction is true. Policy re-evaluation, model re-execution, external effect execution and independent effect verification are false. Destination observation is `not_performed` or `producer_reported`.

DerivationMetadata records the `redacted_derivative` relationship to a source bundle ID.

IntegrityMetadata records a SHA-256 content digest or an HMAC-SHA256 value, with canonicalization profile, subject bundle ID, optional caller-supplied key label and an explicit verification claim.

7 Action Types

The 0.1 format classifies proposals with a deliberately simple set of nine action types:

<code>read</code>	<code>write</code>	<code>external_send</code>
<code>delete</code>	<code>export</code>	<code>purchase</code>
<code>approve</code>	<code>escalate</code>	<code>other</code>

They are not an enterprise ontology. They provide enough structure for run review, validation and governance evidence.

Reconstruction Bundle 0.2.0 does not impose its own action taxonomy. Imported proposals keep the producer's fields, such as the manifest identifiers, target, amount, unit and payload of a bounded refund operation. The importer does not translate them into the 0.1 vocabulary.

8 Bundle Status

The two formats use different status vocabularies, and they must not be mapped onto each other.

Format	Status	Meaning
0.1	complete	package intended to represent a completed run
0.1	partial	package incomplete or still in progress
0.1	redacted	package redacted for export or review
0.2.0	reconstruction_complete	the declared import inputs were completely represented
0.2.0	reconstruction_partial	source dependencies or semantics remain unresolved
0.2.0	incomplete	construction has not established a complete import
0.2.0	redacted	derivative with explicit redaction lineage

In 0.1, a `complete` bundle without final output produces a warning rather than an error. In 0.2.0, every status describes the reconstruction, never the real-world outcome. A fully retained rejected observation, an unknown acknowledgement or a partial destination effect can all be completely reconstructed. Missing dependencies or unmapped fields make the import partial.

9 Producer Import and Revision Pinning

9.1 Supported Producer Profiles

Each importer is pinned to exact producer revisions. The 0.2.0 release added profiles for the legacy Control Plane replay export, the Control Plane bounded run record, and the Execution Envelope 0.2.0 SQLite export. Later work added executor producer profiles 1.0.0 and 2.0.0, plus the merged producer pair used by the selected hub profile. The machine-readable compatibility matrix records the supported pairs.

Generation	Control Plane	Executor	Status
Historical unversioned	28350065	6b0ba118	historical; revision-pinned decoding only
Producer 1.0.0	28350065	1d308faf	historical; source-asserted provenance retained
Producer 2.0.0, observation repair	2ea9528e	177354e9	historical-supported; exact pair only
Producer 2.0.0, persistence repair	248d8996	177354e9	historical-supported; same wire contract
merged-producers-v1	d3dadee7	c3c3ee71	exact merged pair, with Manifest 46c950be

Every pair is exact. A new executor revision cannot be paired with an older Control Plane, or the reverse, and unknown revisions are rejected. The persistence-repair Control Plane revision changed record-store mechanics (locking, reload, flush and fsync, atomic replacement) without changing the consumed bounded-record wire contract. Replay therefore kept Reconstruction Bundle 0.2.0 and gave that revision its own producer profile. The revision argument selects a decoder contract. It does not authenticate the source repository.

9.2 Mapping Without Renaming

The legacy Control Plane serializer's arrays of actions, decisions, policy traces, authority records, reliance records and blocked actions become ordered source records, and the producer's own field names are preserved inside each record. Renaming record content would misrepresent what the original commitment covered. For the bounded workflow, each runtime decision is retained as an immutable record. Each attempt occurrence is retained in source order as a lifecycle transition, and observations and reconciliations

are retained separately. Correlation IDs, manifest identifiers, decision and effect IDs, grant revisions, policy versions, requirement commitments and role-mapping versions are kept where supplied. Blocked and held decisions keep their producer reasons.

9.3 Authority Context Identifiers

Two different values are intentionally kept apart. One is the proposal's `authority_context_ref`, a profile reference. The other is the Control Plane authorization binding's `authority_context_id`, a resolved context-instance identifier. Where the executor carries the profile reference under a similarly named field, reconstruction labels it as a profile reference rather than converting it into the instance identifier. Values that the Control Plane binding does not persist, such as institution and authority domain, are not backfilled from caller assertions.

10 Identity, Lineage and Recovery History

10.1 Cross-Record Chain

The bounded importer validates supplied evidence as one chain. The full runtime proposal commitment must equal the decision binding, and the applicable proposal fields must equal the binding. The Execution Envelope operation must equal the proposal and the grant and effect-limit binding. Durable destination rows must equal the committed operation for grant, target, amount, unit and decoded payload. A copied operation digest is not enough when row content differs. Effect identity is checked against the Control Plane's derivation from proposal commitment, grant ID and grant revision. Conflicting content under an immutable decision, attempt or effect identity is rejected. The adapter accepts one supplied proposal and execution operation, and rejects multiple authorization bindings instead of forcing one to match.

10.2 Attempt Namespaces

The Control Plane and the executor each issue attempt identifiers, and the two namespaces are never treated as equivalent. A link is created only when producer evidence explicitly supplies it, for example through an adapter acknowledgement that carries the executor attempt ID. A Control Plane attempt is accepted only when the complete supplied attempt object matches the retained Control Plane record for that attempt, decision and effect. Execution-result attempt IDs are resolved with their namespace in mind. Dangling IDs and incompatible namespace and status combinations are rejected.

10.3 Observations and Reconciliation

Under producer profile 2.0.0 the importer retains the actual acknowledgement and `newly_executed` flag, even when the observation is null and the observed state is unknown. A null observation confers no absence, rollback, verification or retry claim. Rejected observations, for example stale, malformed, contradictory or wrong-effect evidence, are retained with their content unmodified and with no accepted lineage identifiers. Copied executor reconciliation content must exactly match a supplied retained Control Plane reconciliation. Matching copies establish consistency only, not independent evidence. `observed_absent` is preserved with `retry_eligible=false`. Observed absence grants no permission to dispatch.

For each exact effect, `metadata.effect_observation_history` separates several things: rejected reconciliation indices; the latest accepted observation and the latest supported destination state; acknowledgement lifecycle entries in their own array order; and the latest recorded `retry_eligible` value. An unknown attempt status is never rewritten as a received acknowledgement. Ordering is scoped to each

producer array. Cross-sequence order is recorded as `not_established`, and freshness at import as `not_evaluated`. These are recorded observations, not a new destination query.

10.4 Optional Recovery Inspection

A separate module, recovery inspection API 0.1, was merged after the hub’s selected Replay revision and is not part of the selected release. It reads supplied synthetic recovery-resolution evidence from one exact Execution Runtime revision. The caller must supply the original effect, attempt, decision, grant and grant revision independently, and the module compares them with retained originals, dispositions, receipts and observations. It retains the whole input and checks revision ordering, envelope digests and conflict holds. Where an event was reused with changed content, it reports that the original payload is unavailable rather than reconstructing it. Agreement with the caller’s identity is consistency, not authentication. The module does not open databases, admit observations, retry effects or contact a destination.

11 Semantic Validation

11.1 Format 0.1 Validation

The 0.1 validator emits a `ValidationReport`. Errors invalidate the bundle; warnings indicate review concerns but do not.

Codes	Error condition
E001–E006	bundle, version, run, frame, actor and policy-version identifiers must be non-empty
E007, E008, E010, E013, E014, E016	every proposal, decision, blocked action, authority record, reliance record and trace must carry the bundle’s <code>run_id</code>
E009, E011, E017	decisions, blocked actions and traces must reference an existing proposal
E012	every blocked action must correspond to a <code>block</code> decision for the same action
E015	a reliance record’s referenced action, when present, must exist
E018	a decision’s <code>trace_id</code> , when present, must reference an existing trace
E019, E020	a linked trace’s fingerprint and final result must match the decision
E021	reserved for complete-bundle final-output checks
E022, E023	signed metadata must include algorithm and signature
E024, E025	redacted metadata must include a timestamp and non-empty field list

Warnings W001–W013 flag missing proposals, decisions, traces, reliance or authority records; empty allowed or blocked tool lists; proposals without reasons; decisions without traces; a `complete` bundle without final output; potentially sensitive payload keys without redaction metadata; missing signature metadata; and an embedded validation report whose `checked_id` does not match the bundle.

11.2 Reconstruction Import Checks

Reconstruction Bundle 0.2.0 does not reuse the 0.1 rule table. Its checks are applied at import, and a failure raises an import contract error rather than returning a misleading artifact. They include: exact producer profile and revision selection; the cross-record chain of Section 10.1; immutable-identity conflict checks; attempt-namespaces resolution; observation binding, content agreement and recorded temporal policy for accepted observations; and the proposal payload, execution-envelope payload and effect-identity commitment

checks the importer knows at the pinned revision. Other source digests are retained as attributed claims. Missing optional evidence makes the reconstruction explicitly partial; contradictory supplied evidence is a contract error.

Validation of either kind remains implementation-level. It does not prove that the records are true, that the run was complete, that policy was sufficient or that compliance obligations were satisfied.

12 Redaction

12.1 Format 0.1 Redaction

The 0.1 redaction helper returns a deep copy and never mutates the original. By default it redacts action payloads. Optionally it also redacts targets, the final output, and reason fields on proposals, decisions, rule evaluations and blocked actions. It preserves IDs, timestamps, decision results, policy names, deterministic fingerprints and cross-references. It sets status to `redacted` and records the redacted field paths, timestamp, policy and replacement value. Fingerprints are deliberately not recomputed, so they continue to match the original policy inputs.

12.2 Reconstruction Redaction

Redaction of a 0.2.0 reconstruction follows a fixed order:

1. copy the source reconstruction;
2. assign a new bundle identity;
3. record the `redacted_derivative` relationship to the source bundle ID;
4. modify the selected source records;
5. remove the source integrity metadata;
6. downgrade affected original commitment checks to attributed claims;
7. compute fresh integrity metadata for the derivative.

A stale HMAC or source digest is never presented as authenticating modified content. Redaction in either format is not a guarantee that every sensitive field has been removed. It must be configured and reviewed under the relevant data-handling policy. The helpers are reference implementations, not data-loss-prevention systems.

13 Signing

Both formats support HMAC-SHA256 export integrity with a caller-supplied shared secret. Signing helps a holder of the secret detect modification of an exported bundle. It is not production key management and does not provide a complete trust framework.

13.1 Format 0.1 Canonical Serialization

The 0.1 signing helper dumps the bundle in JSON mode, removes the entire embedded signature metadata field, and serializes the result with sorted keys, compact separators and ASCII-safe encoding. It returns a signed wrapper (`SignedReplayBundle`) whose own signature metadata is authoritative. Embedded signature state on the bundle does not affect signing or verification.

13.2 Reconstruction HMAC

The 0.2.0 HMAC is computed over canonical reconstruction content together with its integrity metadata, with only the HMAC value itself blanked to avoid a circular input. The authenticated metadata therefore includes the HMAC kind, algorithm, canonicalization profile, subject bundle ID, the caller-supplied `key_id` label and the verification claim. Verification also requires the declared subject to equal the actual bundle ID, and the algorithm and profile to match the implemented profile. HMACs produced before this hardening excluded the integrity array from the MAC input. They need explicit legacy handling or re-signing from trusted source content. A content digest (SHA-256) can be recorded separately.

13.3 Limitations

HMAC-SHA256 with a shared secret provides export integrity, assuming the secret is handled securely. It does not provide:

- non-repudiation or a publicly verifiable issuer signature;
- a public-key trust hierarchy or certificate-authority semantics;
- protection against compromise of the signing secret;
- authentication of key identity or custody: `key_id` is an authenticated label, not proof;
- production key rotation;
- regulatory acceptance by itself;
- authorization of any kind.

High-assurance deployments should use appropriate key management systems and may consider asymmetric signing with an HSM or KMS.

14 Command-Line Interface

The package name (`agent-replay-bundle`) and the `arb` command are unchanged by the October 2026 repository rename. The CLI operates on 0.1 bundles.

14.1 Implemented Commands

```
arb validate BUNDLE.json
arb summarize BUNDLE.json
arb redact BUNDLE.json --out REDACTED.json [--targets]
    [--final-output] [--reasons] [--replacement TEXT] [--policy NAME]
arb sign BUNDLE.json --secret SECRET --out SIGNED.json [--key-id KEY_ID]
arb verify SIGNED.json --secret SECRET
arb check-examples
```

The `validate` command exits with 0 for a valid bundle, 1 for an invalid one, and 2 for a file, parse or model error. Reconstruction import, redaction and signing are exposed through the Python API rather than the CLI, including:

- `import_legacy_control_plane_replay` and `import_bounded_workflow`;
- `redact_reconstruction_bundle`;

- `sign_reconstruction_bundle` and `verify_reconstruction_hmac`.

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

A first local check from a repository checkout, with Python 3.11 or later in an activated virtual environment, is:

```
python -m pip install -e ".[dev]"
arb validate examples/customer_service_replay_bundle.json
arb summarize examples/customer_service_replay_bundle.json
arb --help
```

The full selected integration is reproduced from the hub, whose runner supplies the exact producer checkouts and test wiring.

15 JSON Schemas and Public Conformance

The repository includes JSON Schema Draft 2020-12 files for both formats:

- `agent_replay_bundle.schema.json`, the primary schema for the 0.1 objects, with one schema per component object;
- `reconstruction_bundle_0_2.schema.json`, for Reconstruction Bundle 0.2.0.

Schema `$id` values were not rewritten by the repository rename; the 0.2.0 schema keeps the pre-rename repository spelling.

The schemas serve four purposes. They support interchange, so that external systems can understand record shape. They support structural validation outside the Python package. They document the public boundary explicitly. And they provide a basis for future compatibility tests across implementations.

The schema boundary is deliberately limited. Schemas validate structure, required fields, types and allowed values. They do not prove that a bundle is operationally complete, that runtime controls were sufficient or that a compliance obligation has been satisfied. For producer imports, schema validity is also not compatibility: a record that matches the schema but comes from an unsupported revision is still rejected.

16 Example Bundles

16.1 Format 0.1 Examples

The original examples remain. In the customer-service example, a CRM read is allowed and an outbound email is blocked for lack of external-send authority. In the internal-research example, document search and file reads are allowed and an export is escalated. In the procurement example, a purchase-order draft is allowed while approval and vendor notification are blocked. A redacted version of the customer-service bundle is included. So is a signed example, whose signature is a demonstration placeholder that shows structure, not a real signing operation.

16.2 Reconstruction Examples

Two root-level bounded examples, a success reconstruction and a lost-acknowledgement reconstruction, are retained as explicitly historical 0.2.0 artifacts. The `examples/producer-v2/` directory contains reconstructions generated from the exact producer-2.0.0 observation-repair pair (2ea9528e/177354e9),

not the selected `merged-producers-v1` pair. They cover ordinary success, a rejected wrong-effect observation, unavailable observation, restart recovery and prior observed absence. They are produced by the repository's generator scripts, and no digest was edited by hand.

17 Relationship to Producers

Replay Bundle is not a runtime control layer. It depends on records produced by other systems.

- The **Control Plane** supplies runtime proposals, decisions, decision bindings, effects, attempts, observations and reconciliations. Its current role includes effect-time revalidation and persistent record transactions on a single host.
- The **Execution Runtime** supplies the Execution Envelope 0.2.0 operation, destination effects, attempts, append-only attempt events and observations under a versioned producer profile.
- The **Action Manifest** supplies the declared action surface that proposals bind to. The bounded adapter is tested with Manifest 1.1 at its pinned revision and Authority Context 0.1.0.

Replay records only what these producers retained, at the assurance level each producer generation supplied. Historical artifacts never acquire tenant, refusal, stop or adapter assurance by inference or backfill. A prepared contract for future tenant-aware producer generations reserves mappings for tenant binding, failure classes, stop and intervention transitions, and observation finality. It advances no schema version until actual accepted producer output exists. Two related projections are merged in the repository but not selected by the hub: a bounded failure-history projection for one exact Runtime revision, and a source-row authority and stop-lineage importer. Other systems can still produce 0.1 bundles by implementing the public format.

18 Relationship to Governance Evidence Pack and ODES

The **Governance Evidence Pack** consumes Reconstruction Bundle 0.2.0 through a versioned traceable importer (`agep-manifest-reconstruction-import/0.3.2` for the merged producer pair). That importer combines the reconstruction with the Action Manifest and records the derivation, so each summarized decision or outcome can be followed back to its source. Evidence Pack validation does not change Replay's assurance boundary.

ODES, the Open Decision Evidence Standard, is a separate open, vendor-neutral discussion draft with reference tooling for portable decision evidence. Its v0.2 narrative and implementation profile retain the `pder-v0.1` record schema. Replay Bundle is not ODES and does not define ODES conformance. Since July, the ODES repository has implemented its own documented mapping from exact supported Manifest and Replay inputs into decision-evidence packages. That mapping lives in ODES, not in Replay Bundle.

The recommended boundary is:

- Replay Bundle remains a native reconstruction artifact for AI-agent runs.
- ODES remains a vendor-neutral discussion-draft standard for portable decision evidence, evaluated by each recipient under its own policy.
- ODES tooling may export from supported Replay inputs. Package validity does not supply missing authentication or current-authority evidence.
- No conformance to ODES should be claimed unless an ODES profile and conformance test suite define that claim.

19 Security Considerations

Replay Bundle improves reconstructability and evidence handling, but it is not a security product. Important limitations include:

- the bundle records evidence but does not enforce policy;
- import does not re-evaluate policy, renew authority or perform effects;
- reconstruction completeness is not destination finality or delivery verification;
- authority records preserve asserted context but do not grant authority;
- reliance records document sources but do not prove those sources were correct;
- traces record rule evaluation but do not prove policy sufficiency;
- revision selection identifies a decoder contract and does not authenticate repository origin; source-asserted provenance remains attributed;
- synthetic SQLite observation is producer and local destination evidence, not independent institutional verification;
- redaction helpers must be configured and reviewed correctly;
- signing provides shared-secret export integrity, not production key management;
- validation checks structure and consistency, not truth or completeness;
- production deployments still require authentication, authorization, monitoring, incident response, run-time control and organizational governance.

The correct security posture is layered. Replay Bundle supplies structured reconstruction evidence. It should be combined with application authorization, runtime policy gates, secure logging, monitoring, redaction workflows, key management, operational controls and governance review.

20 Implementation Boundary

The public reference implementation is intentionally thin. It provides enough structure to demonstrate the reconstruction category and to support bounded integration qualification. The stack's open-source boundary places record formats, schemas, validators, reference behavior, examples, tests and business-facing documentation in the public layer. Hosted services, integrations, enterprise storage, identity integration, approval workflows, compliance mappings, replay viewers, evidence registries, dashboards, managed policy workflows and customer-specific support may belong to a commercial layer.

Public reference layer	Possible product layer
0.1 replay schema and object model	hosted replay viewers
Reconstruction Bundle 0.2.0 schema and model	enterprise replay storage
Revision-pinned producer importers	integrations with other producers
Semantic validator and import checks	compliance mappings
Redaction helpers	production redaction workflows
HMAC integrity helpers	production key management
CLI utilities	platform APIs and managed services
Example and generated bundles	deployment-specific templates
Tests and qualification records	customer-specific support

Public software exposes the integration contract: what run evidence can be packaged or imported, how it is validated, how sensitive fields can be redacted, and how export integrity is represented.

21 Reference Implementation Status

The repository is <https://github.com/cogno-us/cognous-replay-bundle>, package version 0.2.0, Apache-2.0. The table summarizes merged capabilities. Merged is not the same as selected: the hub lock selects this component at the revision that exposes the Reconstruction Bundle 0.2.0 interface with merged-producer compatibility. Later merged work is accepted in the repository but not selected.

Capability	Status	Notes
0.1 object model, loader, dumper	Implemented	Pydantic v2; unchanged for compatibility
0.1 semantic validation	Implemented	E001–E025, W001–W013
0.1 redaction and HMAC signing	Implemented	signed wrapper authoritative
CLI (<code>arb</code>)	Implemented	validate, summarize, redact, sign, verify, check-examples
Reconstruction Bundle 0.2.0 model	Implemented	separate API and schema
Legacy Control Plane import	Implemented	pinned historical revision
Bounded workflow import	Implemented	cross-record chain validation
Executor producer 1.0.0 and 2.0.0	Implemented	exact revision pairs only
Merged producer pair	Implemented, selected	<code>merged-producers-v1</code> pair
Observation history and loss reports	Implemented	unknown, rejected and absent retained
Reconstruction redaction and HMAC	Implemented	derivative identity, fresh integrity
W3 failure-history and C8 lineage projections	Merged, not selected	exact Runtime revision
Recovery inspection API 0.1	Merged, not selected	optional synthetic module
Tenant-aware producer mapping	Prepared contract	no accepted producer yet
Hosted replay viewer, registry, key management	Not implemented	product-layer concerns
Atomic-claim and refund-intent evidence import	Not implemented	requires versioned producer exports

22 Test Suite and Continuous Integration

Tests are not part of the format, but they are part of the public trust surface: they show which claims are exercised by executable checks. Repository CI runs the test suite on Python 3.11 and 3.12. It checks out the pinned producer revisions it imports from, so each importer is exercised against actual producer code. Separate workflows run the merged-producer qualification and the recovery-inspection qualification against their exact producer revisions.

The suites cover 0.1 loading, model and semantic validation, redaction, signing, CLI behavior and examples. They also cover reconstruction import from real producer output. The covered scenarios are ordinary success, rejected and unavailable observations, restart recovery, lost acknowledgement, partial delivery, prior observed absence with retry disabled, denial without effect, and historical observations. Adversarial cases cover unsupported revision pairs, identity and attempt substitution, copied or substituted reconciliations, promotion of rejected to accepted evidence, retry claims, destination content substitution and HMAC metadata substitution. The generators verify that import leaves the Control Plane record bytes and the logical destination database unchanged. The merged-producer qualification records 19 local cases: 14 producer scenarios plus five unsupported-pair and contradiction cases.

Stack-level evidence is reported by the hub, not per component. In the current release evidence, full candidate CI run 37694032916 resolved all 35 acceptance scenarios with the required suites executed twice. A fresh reproduction on 13 October 2026 ran 1,009 Python testcases in each of two repetitions plus 20 separate Node tests, with 34 passed invariants and one characterized limitation. The optional same-host C1 authority and effect profile passed 73 tests in a separate run and is not part of that count. An earlier 915-test persistence-generation figure is historical evidence for its own revisions and is not current. None of these counts establishes production readiness.

23 Example Use Cases

23.1 Agent Run Review

A platform team reviews a bounded refund run. The reconstruction shows the proposal, the decision and its binding, the executor attempt, the destination observation and the source of each record. Reviewers can reconstruct the run without searching raw logs.

23.2 Incident Investigation

A refund request timed out. The reconstruction preserves the unknown acknowledgement, any rejected observations and the later reconciliation of the original effect. Investigators can establish what was known at each step and confirm that no replacement effect was authorized by the timeout.

23.3 Audit Preparation

An audit team needs evidence that runtime controls operated. Reconstructions show decisions, refusals, attempts, observations, import reports and integrity metadata, each attributed to its producer revision.

23.4 Evidence Pack Generation

A governance team prepares a business-facing evidence pack. The Evidence Pack importer combines the reconstruction with the Action Manifest, and each summary remains traceable to its source.

23.5 Controlled Export

A reconstruction must be shared outside the engineering team. A redacted derivative with its own identity and lineage is produced and signed afresh, while the original remains intact.

24 Evaluation Criteria

The usefulness of Replay Bundle should be evaluated against operational criteria rather than benchmark scores. A deployment should ask:

1. Does the bundle identify the run and the exact producer revisions it came from?
2. Does it preserve proposals, including those never executed?
3. Does it record decisions and link them to proposals and bindings?
4. Does it preserve blocked or refused actions with their reasons?
5. Does it keep the Control Plane and executor attempt namespaces distinct?
6. Does it retain unknown acknowledgements and rejected or unavailable observations?
7. Does it separate reconstruction completeness from delivery outcome?
8. Does it report missing, unmapped and unsupported information?
9. Does it reject unsupported producer revisions rather than guessing?
10. Does it record authority context and reliance as supplied, without backfill?
11. Can it be semantically validated?
12. Can it be redacted without breaking governance structure or reusing stale integrity data?
13. Can it be signed and verified for export integrity, with stated limits?
14. Does import leave producer records and destinations unchanged?
15. Can it support business-facing evidence generation and portable decision-evidence export?
16. Which checks were actually executed in the intended environment?

25 Roadmap

The public roadmap should remain implementation-focused and avoid overclaiming. Since July, several items listed as near-term have been realized in a bounded form: import from Control Plane records, export to the Governance Evidence Pack (through the Evidence Pack's own importer) and an ODES mapping (implemented in the ODES repository). Formal versioning with migration guidance now exists as the separate Reconstruction Bundle 0.2.0 model.

Gated work includes:

- a new producer generation for tenant binding, refusal and stop handling, and observation finality, once accepted producer output exists;
- versioned import of atomic-claim, grant-budget and refund-intent evidence, which requires explicitly versioned producer exports;

- mapping of recovery-inspection output into the Evidence Pack, and composed qualification;
- hub selection of later merged Replay revisions, which is a separate decision from component acceptance.

Possible extensions still listed in the repository include a conformance test suite, bundle diffing, richer redaction profiles, an optional OpenTelemetry mapping and further framework examples. Longer-term work, such as replay viewers, hosted validation, storage adapters and compliance export mappings, belongs to the product layer.

At the stack level, feature intake was frozen on 13 October 2026 for release closeout. Public preview approval is pending, and final publication and operational trust remain on hold. The reference release keeps its boundaries. C0, the default, is pre-effect revalidation and not destination-commit atomicity, so a check-to-commit race remains. C1 is an optional same-host profile that is not enabled by default. No C2 or C3 claim is made. Effect-ID deduplication is not business-intent deduplication: a refund-intent profile exists in the selected runtime, but the default path does not enable it.

26 Conclusion

AI-agent governance requires reconstructability. As agents propose consequential actions, organizations need portable evidence packages that preserve proposals, decisions, attempts, observations, authority and reliance context, validation status, redaction lineage and integrity metadata. Those packages must also keep the uncertainty that real execution leaves behind.

Cognous Replay Bundle provides a minimal public reference for that responsibility in the Replay & Evidence group of the Open Control Stack. It retains the original replay-bundle format and its tools. It adds Reconstruction Bundle 0.2.0, which imports exactly the producer revisions it was qualified against, preserves identities and attempt namespaces, keeps unknown and rejected evidence visible, and reports its own gaps.

The contribution is deliberately narrow. Replay Bundle does not run agents, enforce or re-evaluate policy, renew authority, verify delivery, certify compliance or prove correctness. It supplies a concrete, non-effecting reconstruction artifact for one operational need: making AI-agent runs reconstructable, portable, redaction-ready and integrity-checkable, and usable as supporting evidence for governed deployment review.

References

- Cognous. *Cognous Replay Bundle: Portable reconstruction records for AI-agent runs*. Public reference implementation, 2026. <https://github.com/cogno-us/cognous-replay-bundle>
- Cognous. *Cognous Open Control Stack*. Hub repository, release status and component lock, 2026. <https://github.com/cogno-us/cognous-open-control-stack>
- Cognous. *Cognous Action Manifest*. Public reference implementation, 2026. <https://github.com/cogno-us/cognous-action-manifest>
- Cognous. *Cognous Control Plane*. Public reference implementation, 2026. <https://github.com/cogno-us/cognous-control-plane>
- Cognous. *Cognous Execution Runtime (Molibot Safe)*. Public reference implementation, 2026. <https://github.com/cogno-us/cognous-execution-runtime>
- Cognous. *Cognous Governance Evidence Pack*. Public reference implementation, 2026. <https://github.com/cogno-us/cognous-governance-evidence-pack>

- Open Decision Evidence Standard. Discussion draft and reference tooling, 2026. <https://github.com/cogno-us/open-decision-evidence-standard>
- National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework*. NIST AI 100-1, 2023.
- OWASP Foundation. *OWASP Top 10 for Large Language Model Applications*. Project documentation.
- Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.