PRIVATE VALIDATION · PUBLIC EVIDENCE

CALYBRIS CORE

0.5.7

Adversarial validation recorded by Calyra Engine

RECORDED · 23 JULY 2026

Declared evidence recorded.
Release verification incomplete.

The declared smoke surface was executed against the exact recorded Calybris worktree. No “Verified” badge was issued.

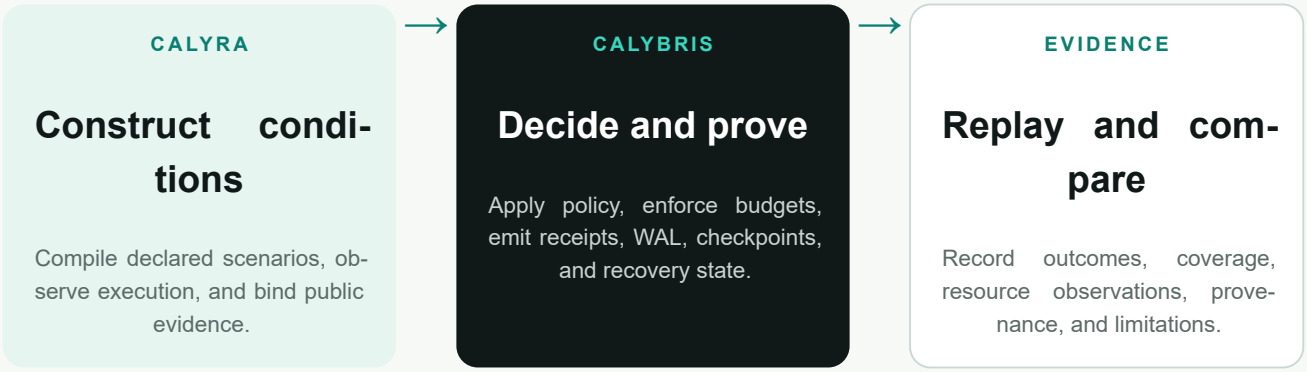
48 / 49 SCENARIOS PASSED	**100%** REPLAY EQUALITY
100% RUST / PYTHON PARITY	**0** FALSE CRYPTO ACCEPTS

VERDICT

The observed decision, replay, parity, and evidence contracts held across the declared run. The release gate correctly remained closed because the target worktree was dirty and the Windows sandbox was not release-safe.

One system creates pressure. The other must remain correct.

Calyra exists to turn an abstract assurance claim into a recorded execution. It is a private adversarial validation and evidence engine—not a component of Calybris Core and not an organizationally independent auditor.



Calyra owns the environment.

It controls the declared test envelope and records what happened. Its proprietary scenario and orchestration details are intentionally absent from this public note.

Calybris owns the answer.

The target must make the decision and produce proof-carrying state. The harness does not replace or “grade on behalf of” the decision kernel.

PUBLIC DISCLOSURE BOUNDARY

This report discloses guarantee categories, aggregate measurements, provenance, and limitations. It does not disclose private fixtures, seeds, schedules, minimized reproducers, injection mechanics, signing material, or Calyra’s internal algorithms.

Nine declared categories. One fail-closed result.

Coverage means every declared invariant and fault point in this recorded pack was exercised. It does not extend to undeclared behavior.

Determinism

Equivalent inputs and state produce the same normalized decision.

Replayability

Recorded proof state can be replayed to the same semantic result.

Fail-closed behavior

Missing, malformed, or unverifiable evidence cannot become an accepted decision.

Budget conservation

Reservations, commits, releases, and recovered totals preserve declared invariants.

Receipt integrity

Decision receipts remain bound to the decision and its proof context.

WAL continuity

Sequence, chain, anchor, and suffix integrity remain verifiable.

Checkpoint & recovery

Persisted state and recovery outcomes respect the declared contract.

Rust / Python parity

The exact native Python build agrees with the Rust target on golden decisions.

Release integrity

Identity, package contents, production hooks, and release prerequisites are checked.

100%

INVARIANT COVERAGE

100%

FAULT COVERAGE

0

FALSE CRYPTO ACCEPTS

INTERPRETATION

A category is not credited merely because the overall scenario passed. Calyra records category-specific evidence and rejects missing or structurally invalid evaluation output.

What the 23 July smoke run actually measured.

49	48	1
DECLARED SCENARIOS	PASSING SCENARIOS	RELEASE-GATE NON-PASS
100%	**100%**	**0**
REPLAY EQUALITY	LANGUAGE PARITY	PRODUCTION HOOKS

RESOURCE OBSERVATION · WINDOWS X86_64

Peak RSS	74.75 MB	CPU time	26.08 s
Disk written	11.38 MB	Wall time	26.81 s
Artifacts read	45.43 MB	Open-file peak	535
Stdout / stderr	124.5 KB / 0 B	Process peak	9

MEASUREMENT NOTE

Resource values are execution evidence, not semantic pass criteria. They vary with platform and host load. The run remained below its recorded limits, but memory and filesystem access were monitored rather than OS-sandboxed.

The result is attached to an exact target—not just a version string.

1. **Target identity**

   Commit, tree, Cargo.lock, platform, and toolchain establish the recorded build context.

2. **Adapter capability**

   A versioned capability manifest states which Calybris surfaces the pack may exercise.

3. **Execution evidence**

   Scenario outcomes, invariant evidence, resources, replay results, and limitations are hash-bound.

4. **Public result**

   Only allowlisted aggregate evidence leaves the private evidence tree.

5. **Release gate**

   A verified status requires every mandatory gate, trusted signature, and a release-safe enforcement backend.

RECORDED PROVENANCE

COMMIT	`63163e46a3b7e5f4a44e7a7a182d406e1372da55`
TREE	`f6831b0ba46f249e767806477da6bf5755dea4f9`
CARGO.LOCK SHA-256	`b7738142bf3fa6aeb396324df1adff29ac23ab5d4348e7029f82ffc907e01026`
ADAPTER DIGEST	`59abc8e0b1d2db22387085bafca152af52b14491b1c7a78818ee653e753d43af`
PACK DIGEST	`e10e417206615abe19952798d53247ab6a908ad17b3b4600324eddf473244617`

EXACT LANGUAGE BINDING

The Python parity gate binds the installed native module to the same commit, tree, lockfile, dirty flag, and full-worktree source digest as the Rust adapter. A same-version wheel built from different source is rejected.

Strong evidence is not the same as a release attestation.

HELD IN THIS RUN

Decision and evidence contracts

- 100% declared invariant coverage
- 100% declared fault coverage
- 100% replay equality
- 100% Rust / Python parity
- zero cryptographic false accepts
- zero production hook symbols

GATE REMAINED CLOSED

Verification prerequisites

- target worktree was dirty
- memory enforcement was not OS-backed
- filesystem isolation was not OS-backed
- enforcement backend reported release_safe=false
- no verified badge or signed release was produced

WHY 48 / 49 IS THE RIGHT RESULT

The sole non-pass was `release.clean_git`. The audited worktree intentionally contained the 0.5.7 release-candidate fixes documented here. Treating that state as verified would weaken the release contract; rejecting it confirms the gate is fail-closed.

No badge was minted.

Calyra's public result remains `verification_status: incomplete`. A future verified run must target an exact committed tree and use an enforcement backend that satisfies the release-safe policy.

Read the evidence narrowly. Use it seriously.

What this note supports

For the exact recorded 0.5.7 worktree, the declared smoke pack exercised its published assurance categories, reproduced semantic outcomes, matched Rust and Python decisions, and found no production hook symbols or cryptographic false acceptance.

Sanitized public method

Calyra identifies the target, checks adapter capabilities, executes the declared pack under bounded resources, validates evidence contracts, replays outcomes, measures language parity, and emits only allowlisted aggregate evidence.

What it does not support

It does not prove the absence of defects, cover undeclared scenarios, replace an external security audit, certify deployment configuration, or grant a Calyra Verified release status.

Proprietary boundary

Private scenarios, fixtures, seeds, schedules, minimized reproducers, fault mechanics, orchestration, scoring, sandbox internals, and signing material are intentionally excluded.

Release-candidate evidence: strong.

Verified release status: correctly withheld.

The next meaningful proof point is a clean-tree run on a release-safe enforcement backend—not a more optimistic label on the current result.

PUBLIC REFERENCES

github.com/emirhuseynrmx/calybris-core
Sanitized Calyra result · JSON

Document version 1.0 · Evidence date 23 July 2026 · English public edition