Every Result Needs a Trail

A practical architecture for reproducible computational experiments

LetX Research Engineering

Technical Presentation

August 2026

Reproducibility is a chain, not a final checklist

The design question

What must travel with a result so that another person can inspect, replay, and challenge it?

The answer spans the full experiment lifecycle: intent, inputs, execution, evidence, and interpretation.

A saved script does not preserve the experiment

- Parameters drift between exploratory runs and reported results.
- Data snapshots change while filenames remain the same.
- Package, hardware, and operating-system details disappear.
- Manual exclusions survive only in a notebook margin or chat thread.

The hidden failure

The chart can remain visible even after the path that produced it has been lost.

The required shift

Treat provenance as part of the result, not as documentation added later.

Three principles keep the evidence attached

Capture by default

Record configuration, code identity, data identity, environment, and seed at execution time.

Make runs immutable

Create a new run record for every change; never revise history in place.

Verify independently

Test replayability and interpretation with checks separate from the analysis code.

The trail should be cheaper to create than to reconstruct.

One manifest connects intent to evidence

Question → **Specification** → **Execution** → **Evidence**

hypothesis and scope | manifest and tests | isolated run | metrics and artifacts

The manifest declares

Inputs, parameters, code revision, environment contract, random seed, expected outputs, and validation rules.

The run record returns

Resolved identities, timestamps, logs, checksums, metrics, outputs, and the verdict of every validation rule.

A run fingerprint makes silent changes visible

For a computational run r, define its identity from the resolved experiment state:

$$\phi(r) = H(h_{\text{code}} \,\|\, h_{\text{data}} \,\|\, h_{\text{env}} \,\|\, \theta \,\|\, s)\,,$$

where θ is the canonical parameter set, s is the random seed, and H is a collision-resistant hash.

Same fingerprint

The declared computational state is identical and the run is eligible for a replay comparison.

Different fingerprint

Something material changed; compare manifests before comparing outcomes.

A fingerprint identifies the declared state; validation still determines whether the run behaved as expected.

The manifest answers the reviewer before the reviewer asks

Artifact	What is captured	Reviewer test
INTENT **Plan**	hypothesis, scope, exclusions, acceptance rule	Does the result answer the declared question?
INPUT **Data**	immutable ID, schema, lineage, license	Can the exact input be recovered?
RUN **Runtime**	locked dependencies, platform, seed	Can the computation replay under contract?
OUTPUT **Evidence**	logs, metrics, artifacts, checksums, tests	Do outputs pass the declared validation?
CLAIM **Claim map**	claim, evidence link, limitations, owner	Is the conclusion supported without hidden steps?

Validation moves from execution to interpretation

1. **Integrity** Inputs and outputs match their checksums.
2. **Executability** The environment rebuilds and the workflow completes.
3. **Determinism** Repeated runs meet the declared tolerance.
4. **Robustness** Controlled perturbations reveal sensitivity.
5. **Traceability** Every claim links to a checked artifact.
6. **Reviewability** A new reader can follow the trail without private context.

Acceptance rule

Publish only when every required rung passes; otherwise place the owned exception beside the claim.

Controls should target the way the trail actually breaks

Failure mode	Observable signal	Control
Mutable input data	Same path, different checksum	Content-addressed snapshots and schema checks
Environment drift	Rebuild changes the dependency graph	Locked dependencies and a captured runtime image
Untracked parameter edit	Metric changes without a code change	Canonical manifest and fingerprint comparison
Selective reporting	Output exists without its rejected alternatives	Append-only run registry and declared selection rule
Unsupported conclusion	Claim has no direct artifact link	Claim-to-evidence review before release

Adopt the trail in four bounded steps

STEP 1

Inventory
Map where inputs, parameters, environments, and outputs currently live.

STEP 2

Standardize
Define one manifest schema and one immutable run directory convention.

STEP 3

Automate
Capture identity and validation evidence inside the execution path.

STEP 4

Govern
Require evidence links at review and rehearse replay with a new operator.

Start with the highest-value workflow

Choose one result that is frequently revised, reused, or challenged. Make its trail complete before scaling the convention across projects.

Before the next run, decide what must be replayable

- Put the experiment contract in a machine-readable manifest.
- Give every run an immutable identity and evidence bundle.
- Validate the trail independently of the analysis that produced it.
- Link each released claim to the artifact that supports it.

The practical standard

A credible result is not merely repeatable by its creator; it is inspectable and replayable by someone new.

Preserve the path, not only the picture.