



UNORYN Litepaper v0.2

Release Candidate 1 | English edition | Review 3 | September 12, 2026

Genesis / Prototype Stage

Proposed Design

RC1 identifies a document release candidate. It does not mean that the protocol has been validated.

UNORYN proposes a task distribution, result verification and contribution provenance workflow for small development teams that need to reproduce checks. The first recommended experiment is a deterministic rule check on offline smart-contract text: a contributor's computer claims a bounded task, returns structured output, and has that output reviewed by a designated verifier before a traceable contribution is recorded.

This complete review draft is for users, developers, contributors and prospective partners. It does not establish that the prototype has run or that the protocol has been audited. The operating-status baseline is the September 11, 2026 handover. By September 12, bilingual static website revision files and documents had been prepared for local preview; this does not confirm that the live site was updated. Public enrollment, a Node implementation, a multi-device demonstration and a Dashboard backed by real events still lack acceptance evidence. The mechanisms below remain recommendations requiring experiments and founder decisions.

1 Executive summary

When analysis or evaluation crosses people, computers and software versions, its context is often lost: which program ran, which data it used, who reviewed it, and whether the current report is still valid after a revision. UNORYN aims to make those relationships queryable, reviewable and exportable so that teams need not rely only on screenshots or verbal assurances.

The long-term vision is a contribution network for shared intelligence, accommodating compute, knowledge, intelligence artifacts and validation. The practical starting point is one bounded workflow, rather than four simultaneous marketplaces. First complete registration, claiming, execution, return, verification, recording and display on one machine; then test disconnection, retries and reproducibility on 2-5 real independent computers. This is an experimental target, not an existing node count.

Contribution Graph would record relationships among tasks, attempts, artifacts and reviews. Verified Delta would record evaluation differences between a candidate and a frozen baseline. Both build on established provenance and evaluation methods. UNORYN must demonstrate that cross-participant coordination reduces review and accountability costs; renaming existing technology is insufficient.

Genesis proposes a centralized Coordinator, an ordinary database and a Python command-line Node. The Coordinator controls participation, task distribution and acceptance, so this is not a trustless system. A native token is unnecessary initially. Points would test contribution accounting and carry no redemption, airdrop or return promise. Continued development depends on real demand, reliability, security and full costs, rather than followers or token valuation.

1.1 Current status and review priorities

Area	Available material	Next evidence required
Website and papers	Bilingual static website revision, Chinese design drafts and corresponding English editions	Deployment confirmation and browser review
Genesis protocol	Task, state, interface and test specifications	Reproducible end-to-end runs and fault-injection records
Demand and economics	Experiment methods and cost formulas	External repeat use, measured costs and payment terms

This revision removes implementation ambiguities: output bytes and locations, current credential and lease eligibility, verification eligibility, frozen policy fields, and protection against deletion of evidence during review and acceptance. The test matrix specifies work to perform; it is not a record of passed system tests.

2 Users and the problem

Potential initial users are small teams maintaining open-source analysis tools, curating public code samples or comparing rule versions. They may need another computer to reproduce a check, isolate a difference and preserve an evidence package for colleagues. Requesters define the question and provide authorized samples. Consumers are maintainers or researchers deciding whether to adopt a tool version. Potential paying customers are teams with continuing needs for reproducible reports and hosted support; Node participants are not assumed to finance the service by buying tokens.

Local scripts, CI services, cloud computing and experiment trackers already perform much of this work. Moving an unchanged script to another person's computer while adding registration, downloads and troubleshooting is not sufficient adoption value. The hypothesis is that a common task and evidence format makes cross-device review and accountability easier.

Switching costs include packaging, permission review, environment adaptation, scheduling, learning formats and handling new failures. A script that takes seconds and is used only by its author does not need UNORYN. A team whose cloud platform already meets its cost, permission and reproducibility needs should not be forced to migrate. A useful open-source evaluation and evidence tool would itself be a valid outcome; a large network is not the only acceptable result.

3 Long-term vision and Genesis scope

“One Planet. One Intelligence. Built by Everyone.” expresses a participation vision. It does not describe an existing global network jointly training a large model. Distributed inference, model evaluation, model patches and joint training have different bandwidth, privacy, synchronization and verification requirements and need separate experiments.

The Genesis deliverable is one fully processed logical task. Multiple retries still count as one success through its main acceptance slot. Additional review tasks are identified separately as validation work. Public statistics must retain failures and pending review rather than count a Node's completion claim as an effective contribution.

This phase does not promise permissionless access, arbitrary third-party code execution, general defenses against malicious nodes, real-time financial decisions, formal smart-contract audits, on-chain revenue distribution or distributed training. A clear list of unsupported inputs and devices reduces development and support burdens.

4 Initial use case and demand validation

4.1 Candidate workloads

Candidate	What it can test	Main burden	Recommendation
Offline contract-text rules	Task loop, version consistency and review	Limited representativeness; no audit capability proof	First Genesis workload
Local small-model classification	Local AI inference and evaluation	Licensing, downloads, hardware and randomness	After the basic loop passes
Real-time crypto intelligence reports	Information synthesis and subjective value	Freshness, source rights, evaluation and misleading conclusions	Defer

The recommended first experiment is an offline rule-sample task-network loop test. Begin with small texts written by the team and explicitly licensed for publication, including matching structures and counterexamples. Third-party samples require individual license checks. Public visibility alone does not authorize copying, training or redistribution.

4.2 Inputs and outputs

The input package contains UTF-8 texts, sample IDs, permissions, a rule version and content digests. Samples are read as data: no contract execution, wallet connection, on-chain transaction or access to real funds. Rules may check defined syntax patterns and metadata completeness. A match means the defined pattern was found, not that a vulnerability exists or that the code is safe.

JSON output identifies sample_id, rule_id, matches, locations and reasons an input could not be handled, plus input, Runner and output-schema versions. Locations are half-open intervals [start_byte, end_byte) in the original UTF-8 bytes. Line endings and Unicode must not be silently normalized. Each (sample_id, rule_id) has

exactly one row; multiple matches belong to that row's ordered array. Unsupported syntax returns UNSUPPORTED, never a negative result. The verifier checks the package and structure, then compares normalized results using independently written reference checks or known answers.

The proposed environment is a frozen Python CPU Runner with read-only inputs, an isolated work directory, no task-time networking, and limits on memory, duration and output. Freeze production-task limits after measuring supported machines; conservative, documented initial limits may support controlled tests. The Runner does not read personal directories, install task-supplied dependencies or download and execute scripts.

These deterministic tasks may be cheap enough that full verification costs more than execution. Their purpose is to simplify protocol debugging, not establish commercial demand. Later, an authorized small local model can classify offline samples in the same task envelope. Record model and tokenizer versions, backend and resource use. On devices controlled by a trusted observer, such evidence may support a local-run claim. Remote self-reported logs do not prove actual execution or exclude outsourcing. Cloud APIs can be a comparison but do not count as local AI compute contributions.

4.3 Low-cost demand experiment

Ask potential users for a check they already need. Compare the total time of their local-script workflow against the evidence-report workflow, including packaging, waiting, troubleshooting, reading and resolving reproduction disputes, not just inference. An initial 3-5 non-team interviews would be exploratory, not a market-size estimate.

A proposed continuation condition is at least two independent users submitting their own authorized tasks again without an airdrop promise and explaining how the report informed an actual decision. A subsequent quotation with defined costs can test willingness to pay. Negative evidence includes interest only in demonstrations, repeat use still requiring team operation, or coordination costs exceeding review value. In that case, narrow the product to a local reporting tool and pause node expansion.

5 Roles and the minimum workflow

The requester defines needs, permissions and acceptance criteria; the contributor operates a Node; the Coordinator matches work and manages leases and states; the verifier applies frozen policy; the consumer reads the report; maintainers handle upgrades, disputes and pauses. One team may initially perform several roles. Disclose this; role counts are not independent participant counts.

The path is registration, capability declaration, lease claim, package and resource checks, local execution, structured submission, verification, acceptance with a contribution record, then Dashboard display. Claiming gives time-limited rights. A new submission must pass server eligibility checks before both execution-lease expiry and the task deadline, and its transaction must commit successfully. A timely submitted result remains under verification after the execution lease expires; it must not cause the main task to be reassigned. Acceptance still requires eligibility, cancellation, pause and revocation checks before the review deadline. Revoked credentials cannot claim or submit anew. Previously submitted results follow Whitepaper Sections 5.1 and 6.2-6.4 and Appendix C.

Task and Attempt have separate IDs. After current authentication, the same idempotency key and content return the existing receipt; reusing the key with different content is rejected. The same Attempt and content under a new key also return the original receipt. A receipt is not proof of current acceptance eligibility. BLOCKED tasks, paused acceptance, expired review and revoked results cannot resume automatically. New submissions separately validate the Attempt's secret lease token, which must not appear in public artifacts. Verification workers must submit while their own lease is valid and their generation is current; expired work cannot change acceptance eligibility.

Acceptance and contribution records are written in one database transaction, with uniqueness constraints preventing duplicate accounting for a slot. This constrains database acceptance, not whole-network exactly-once behavior or repeated compute consumption.

The Dashboard should distinguish logical tasks, attempts, structurally valid submissions, verified results, accepted results, failures and cancellations. Split every metric by Task.task_role; a validation task with an acceptance slot is not new customer demand. Completed is a Node report, Verified means technical checks passed, and Accepted means formal policy-based acceptance. Participants should be able to export evidence, not only view points.

6 Four contribution types

Compute means bounded execution work. Genesis first records which assigned client submitted an accepted result and what supports any actual-execution claim. A valid output alone supports an accepted result submission, not proven compute delivery. Self-reported hardware prices or wall-clock time do not establish value.

Knowledge includes authorized data, labels and expert feedback, with provenance, permissions and versions. More data is not automatically better. Initial sample preparation can be recorded without launching a data market.

Intelligence artifacts include rules, code, analysis methods and model changes. Running someone else's method does not transfer its authorship to the first result submitter. Comparative evaluation is needed before reporting a candidate improvement.

Validation includes reruns, checks and evaluation reports with their own versions and evidence. Majority agreement does not establish genuinely independent execution. Genesis enables controlled submissions and validation records; knowledge and methods are referenced as authorized artifacts, while economic rights are not automatically assigned by the graph.

7 Contribution Graph

The graph connects Contributors, Nodes, Tasks, Attempts, Artifacts, Verifications and Contribution Records. One participant may operate many Nodes and one computer may create many Node IDs. Record devices, operators and network conditions separately: neither different IDs nor different IPs prove independence.

Minimum records include stable IDs, schema versions, input/output digests, environment identifiers, server timestamps, verification policies and references. Artifact derivation can form a directed acyclic graph: a new

version derives from older versions without a cycle. Use, review, dispute and revocation are ordinary references and need not share that DAG constraint.

Identical content can reuse storage while submissions under separate assignments remain separate. This does not prove that different people executed the program. Revisions create new versions pointing to old records. Revocation preserves reasons and history but removes revoked contributions from current effective views. Content duplication, authorship disputes and duplicated execution are different problems; one hash cannot solve all three.

Hashes check consistency with a digest, not correctness, execution, device independence or an administrator-proof history. Signatures authenticate a message to a key, not real-world identity or authorship. Ordinary databases initially rely on administrators and backups. Append-only logs, externally retained digests or later anchoring can improve detectability of changes but cannot repair bad data or missing licenses.

W3C PROV, OpenLineage and MLflow provide established concepts for provenance, run events and experiment tracking. [R7-R9] UNORYN's unproven value is an evidence package connecting delegation, acceptance reasons, revisions and reuse in a form reviewers actually use. If that does not reduce tracking effort relative to existing tools, reuse their formats rather than preserving a custom graph as an end in itself.

8 Verified Delta

Accepting a task means it met its stated requirements. Showing that a candidate outperforms a baseline is a separate comparative evaluation. Repetition may strengthen evidence without creating new intelligence. Verified Delta is an evaluation and review record, not a universal unit of intelligence value.

The first automatic decision procedure is limited to deterministic binary rule outputs on a fixed finite set, as specified in Whitepaper Section 10.6. Freeze the manifest and labels, both versions, environment, F1 thresholds and resource caps. Run each version twice in the specified order. Require complete row coverage, identical repeat outputs, trusted measurements and review records. Public-set conclusions apply only to that set and cannot establish generalization. Report quality, latency and cost separately, with units, directions and limits, rather than conceal tradeoffs in an opaque weighted score.

Absolute quality change is candidate minus baseline. Relative change additionally divides by a nonzero, meaningful baseline. Latency savings can be baseline minus candidate. Random workloads require per-sample, per-run records and uncertainty methods appropriate to sampling units and dependence. Without adequate evidence, report observations and do not invent confidence intervals or significance.

Illustrative/synthetic data, not UNORYN measurements: among 200 binary samples, baseline TP=80, FP=20, FN=20 gives F1=0.8000; candidate TP=86, FP=18, FN=14 gives F1 approximately 0.8431. The absolute increase is about 0.0431, or 4.31 percentage points, and the relative increase about 5.39%. Without a frozen manifest, usage scope, paired outputs and review evidence, these numbers demonstrate arithmetic only. The conclusion remains insufficient evidence.

The four outcomes are improvement, no improvement confirmed, regression and insufficient evidence. Check evidence eligibility first; incomplete evidence permits only the last outcome. With complete evidence, excessive quality loss or any resource-guardrail breach is regression. Otherwise, meeting the minimum gain

and every guardrail establishes improvement within scope; a smaller gain means no improvement confirmed. Statistical decisions for random models remain unsupported in this phase.

Repeated tuning can turn a holdout into development data. Limit access, record all candidates and use new holdouts for confirmation. Inferences about future samples after screening multiple candidates require a preregistered comparison and error-control plan. Fixed public-set comparisons report only that set and retain candidate history.

Combined improvements can reinforce or offset one another. Individual deltas cannot simply be added to divide all rewards. Record joint authors, dependencies and experiments. Where attribution is unreliable, record a joint contribution or defer economic allocation. Benchmark gains do not establish general intelligence, future investment returns or production reliability.

9 Verification and security boundaries

Deterministic tasks favor known answers, independent reference implementations and controlled reruns. Model evaluations need frozen data and explicit tolerances; subjective analysis needs predefined rubrics and reviewers who preserve disagreement. Work that cannot be verified should not be represented as automatically trustworthy.

The initial Coordinator and verifier may share control. The team may restrict access, reorder queues or pause the system; record those actions. Independent computers can reveal some environment differences but do not exclude shared bugs, copied answers or collusion. Hardware, time and identity declarations are diagnostic, not independent proof.

Controlled tests distribute reviewed, bounded Runners and authorized inputs. Public clients additionally require clear consent, least file/network privileges, resource limits, package authenticity checks, version revocation, emergency stop and independent security review. Docker, signatures, staking and majority votes cannot individually make arbitrary third-party code safe.

Replay differs from copying. Binding tasks, attempts, leases and idempotency keys can constrain replay. A correct copied answer without actual execution cannot be detected merely from an output hash. Even noncash points can motivate reputation farming or airdrop speculation. Hidden labels cannot stop input forwarding; copying, shared bugs and outsourcing may remain undetected. Detection rates, false positives and costs have not been measured.

10 Business and economic coordination

The direct token-free path is hosted tasks, reproducible reports and support billed under an explicit service scope. Pricing, charging eligibility and the responsible entity remain undecided. Existing payment channels or settlement assets may later support cross-participant payments, with their own fees, refunds and jurisdictional obligations. On-chain settlement does not require a new asset. Every currently defined function can operate without a native token.

Task settlement, objectively provable breach bonds, resource allocation, compensation and limited governance have alternatives: invoices, applicable ordinary security arrangements, quotas, service revenue or grants, and maintainer governance. These are implementation options, not proof of token necessity. Unclear

errors should not trigger automatic bond deductions; nonexistent scarcity should not create holding requirements; voting cannot replace consent, licenses or technical safety.

Genesis points would measure test contributions under a disclosed policy. They have no cash value, income right or future conversion promise. Accounting must be explainable and correctable without duplicate accumulation. Airdrop expectations must not be used to attract participants and then presented as evidence of demand.

Unit economics must include node compensation, all verification and retries, execution retries, bandwidth, storage, coordination, human support, rework, payment fees and fulfillment remediation. Deduct refunds from revenue for the same service cohort. Accepted primary tasks, billable tasks, net revenue and actual receipts are separate measures; validation tasks do not increase the primary-task denominator. Shared machines cannot supply the same capacity twice, and verification queues also limit throughput.

Count each expense once. Allocate protocol-wide review costs by a rule frozen before the run; unknown allocation is not zero. Allocate fixed costs over effectively accepted primary tasks in that cohort. Low utilization raises unit costs. Report cash and economic costs separately, since volunteer time is not permanently free. No measured verification cost or supported price currently establishes commercial viability. If verification costs more than execution, users must value that review enough to pay; otherwise use hosted batches or local tools. Grants finance experiments without proving sustainability.

Token capitalization is a price-times-quantity valuation and trading volume is cumulative turnover; neither is project cash. Sale proceeds require deductions for costs and restrictions. Liquidity-pool assets may be committed to market-making and cannot simultaneously fund unrestricted development. Without financing, limit tests to existing computers, authorized samples and bounded commitments.

Token name, ticker, chain, supply, allocation, lockups, issuer, jurisdictions, financing and TGE remain undecided. This paper gives no legal issuance opinion; any later activity requires advice appropriate to the actual arrangements and locations.

11 Genesis roadmap

All gates below are proposed, not achievements. The handover's end-of-September 2026 Genesis Ready target and October 1-3 small release window are contingent on testing, not mandatory launch dates or token events.

Stage	Hypothesis and minimum artifact	Evidence and response to failure
G0 Single machine	Task, Runner, verifier and records form one loop	Normal, duplicate, cancel and restart tests; otherwise fix the single-machine path
G1 Independent devices	2-5 real computers reproduce compatible work	Device relationships and explained differences; otherwise restrict compatibility
G2 Real display	Dashboard faithfully represents records	Task tracing, reconciliation and revocation; otherwise keep exports only

Stage	Hypothesis and minimum artifact	Evidence and response to failure
G3 Real demand	External users want to return	Authorized tasks and repeat use without rewards; otherwise narrow to a local tool
G4 Cost and reliability	Verified service is worth using	Full costs, failure rates and user tolerances; otherwise narrow or stop
G5 Openness and economics	Greater openness addresses observed limits	Security, disputes, demand, funding and rights are supportable; otherwise remain controlled and token-free

Each gate needs reviewable evidence, although distinct research can proceed separately. Local AI evaluation does not require P2P; a database migration does not require a token. A two-person team should prioritize bounded protocols and reports and defer automatic updates, open markets, training and on-chain settlement.

12 Risks and participation

The main risks are absent demand, excessive verification cost, copied outputs, shared bugs, insufficient data rights and limited maintenance capacity. Node counts do not prove independence, reproducibility does not prove correctness, and interest does not prove willingness to pay. Remaining a prototype, narrowing scope or stopping network expansion are valid responses.

Potential users can prepare an authorized, low-risk task they already need and identify their existing method and unacceptable errors. Developers can review the protocol and add counterexamples and recovery tests. Testers may join controlled reproduction after resource boundaries are clear. Participation requires no asset purchase. No verified clickable enrollment link is provided. The handover identifies @unorynai and unoryn.com; testing arrangements require later team confirmation.

13 Related work and references

Bitcoin is a precedent for reasoning from a concrete problem through trust, network and incentive assumptions, not a mining template to copy. [R1] GEOD is GEODNET's token; its documentation connects station services, quality and incentives, suggesting that service demand should precede allocation choices. [R2-R3]

Gensyn covers machine-intelligence execution, verification, communication and coordination and documents reproducible execution tools; UNORYN cannot claim exclusive ownership of combining contribution and verification. [R4-R5] Bittensor's subnet utility scoring and incentives also overlap and depend on stated trust assumptions. [R6] io.net documents computing resources, hardware challenges and multiple payment methods; this review claims no measured performance. [R13] Whitepaper Section 2 provides the fuller comparison.

The papers share references R1-R13b. W3C PROV, OpenLineage and MLflow inform provenance and experiment records; Celery informs task retries; SQLite informs transactions; RFC 8785 informs JSON canonicalization. Citations are mechanism precedents, not partnerships, endorsements or proof that UNORYN implements these capabilities.

Source links

Source check: September 12, 2026. Official documentation supports only the described mechanisms, not independent performance or UNORYN implementation. Version numbers identify the reviewed pages. See the accompanying source notes for boundaries.

- R1 [Bitcoin whitepaper \(2008\)](#)
- R2 [GEOD Token Introduction](#)
- R3 [GEODNET station performance](#)
- R4 [The Gensyn Protocol](#)
- R5 [Gensyn REE v0.6.0](#)
- R6 [Bittensor Yuma Consensus](#)
- R7 [W3C PROV Overview \(2013\)](#)
- R8 [OpenLineage Object Model 1.53.0](#)
- R9 [MLflow Tracking](#)
- R10 [Celery Tasks 5.6.3](#)
- R11 [SQLite Transaction](#)
- R12 [RFC 8785 \(2020\)](#)
- R13 [io.net Proof of Work](#)
- R13b [io.net Cloud Payments](#)
- R14 [SQLite Foreign Key Support](#)
- R15 [SQLite Backup API](#)