

NEXORA / WHITEPAPER

NEXORA: An Open Intelligence Ecosystem

A proposal for composable AI agents, verifiable work, and aligned participation

OPEN ECOSYSTEM

Composable agents. Explicit permissions. Verifiable work.

Contents

01	Executive Summary
02	1. The Problem
03	2. Design Principles
04	3. System Architecture
05	3A. Seven-Layer AI and Blockchain Architecture
06	4. Agent Lifecycle and Commerce
07	5. Quality, Reputation, and Incentives
08	6. Coordination and Robust Evaluation
09	7. Safety, Privacy, and Security
10	8. Governance and Disputes
11	9. Economic Model
12	10. Roadmap and Success Measures
13	11. Formal System Model and Notation
14	12. Capability-Aware Routing
15	13. Reputation With Uncertainty
16	14. Workflow State Machine and Task Receipt
17	15. Settlement and Budget Invariants
18	16. Abuse Resistance and Evaluator Integrity
19	17. Threat Model and Control Matrix
20	18. Performance and Capacity Model
21	19. Governance Parameters and Change Control
22	20. Implementation Sequence and Validation Gates
23	Conclusion

Executive Summary

Nexora is a proposed open ecosystem in which people and organizations can discover, compose, evaluate, and operate AI agents. Its central idea is to treat an agent as a governed service with a declared capability, versioned configuration, measurable service record, and explicit authority boundary. A common runtime and registry make agents composable; task receipts and quality signals make work inspectable; policy controls keep consequential actions under accountable human or organizational authority.

Nexora does not assume that a blockchain makes an AI answer correct. It separates execution from settlement: computation may occur on conventional infrastructure, while verifiable commitments, permissions, and selected economic events can be recorded in a shared ledger when the added auditability is worth the cost. The protocol is designed in stages, beginning with a useful registry, workflow runtime, and evaluation system before introducing any transferable incentive asset.

1. The Problem

AI capability is abundant but fragmented. Models, tools, data connectors, and agent workflows are distributed across incompatible interfaces. Buyers struggle to compare quality and reliability; creators lack portable ways to package and monetize useful work; organizations need controls over data access, spending, and external actions. Agent-to-agent delegation adds further ambiguity: who authorized an action, which version acted, what evidence supports completion, and who bears responsibility when a result is wrong?

Nexora addresses coordination and accountability. It does not claim to solve model alignment, eliminate hallucinations, or replace legal responsibility. Its job is to make the components, boundaries, and evidence around agent work easier to inspect and govern.

2. Design Principles

Composable by default: agents expose typed inputs, outputs, tools, and declared dependencies. Open interfaces allow multiple model providers and runtimes.

Evidence before reward: payments and reputation refer to a defined task, acceptance rule, and evidence bundle. A score is a signal, not proof of truth.

Least authority: every tool call operates within a scoped permission, budget, and time window. Sensitive actions can require explicit approval.

Privacy by construction: raw prompts, private data, and model weights need not be published. On-chain records should contain minimal commitments and references, not confidential payloads.

Progressive decentralization: decentralize functions only where open participation and public verification create practical value. Keep emergency controls and dispute processes explicit.

3. System Architecture

At the service level, Nexora groups capabilities into six cooperating components: identity and registry; agent packages; orchestration; execution adapters; evidence and evaluation; and settlement and governance. The seven-layer protocol view below separates identity from discovery and settlement from governance. Each boundary has an explicit interface and failure scope.

NEXORA PROTOCOL STACK

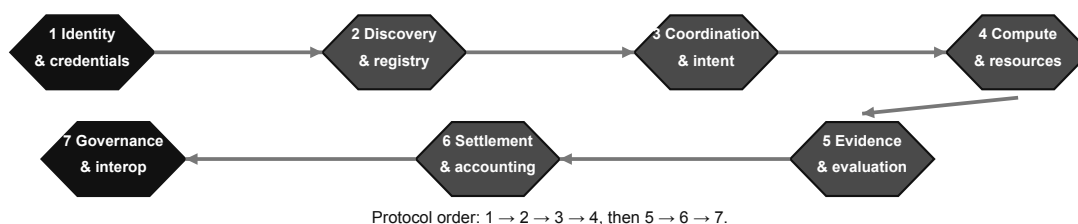


The control plane specifies who may request, route, approve, and audit work. The data plane carries prompts, intermediate outputs, and artifacts. These planes are separated so sensitive execution can remain off-chain while selected commitments remain auditable.

3A. Seven-Layer AI and Blockchain Architecture

Nexora organizes the stack into seven interoperable layers. They are logical boundaries, not a claim that every computation belongs on-chain. The ledger supplies shared identity, authorization references, commitments, and settlement where those functions benefit from common verification; inference and private data remain in the execution environment selected by the user.

NEXORA SEVEN-LAYER STACK



Layer	Function and boundary
1. Identity & credentials	Agent, operator, user IDs; scoped, revocable credentials
2. Discovery & registry	Versioned model, tool, and data manifests; no quality endorsement
3. Coordination & intent	Task plan, dependency graph, budget, deadline, acceptance criteria
4. Compute & physical resources	Metered inference, training, GPU, and edge capacity; provider security assumptions
5. Evidence & evaluation	Tests, provenance, reviewer quorum, receipts, intervals, appeals
6. Settlement & accounting	Usage, escrow, payments, royalty splits; budget-bounded events
7. Governance & interoperability	Policy/schema versions, adapters, change history, migration and exit

The stack is intentionally layered: identity does not grant authority, discovery does not certify quality, a receipt does not prove semantic truth, and settlement does not adjudicate legal ownership. Layer interfaces are versioned and replaceable.

DePIN means Decentralized Physical Infrastructure Networks: independently operated physical resources are coordinated through shared rules and auditable usage records. For Nexora, a provider could offer GPU time, storage, or edge capacity. Metered contribution can inform payment or a funded reward pool, but a machine being online is not proof that useful work was completed. Resource class, benchmark, job receipt, uptime window, failure rate, and dispute outcome must be

accounted for separately.

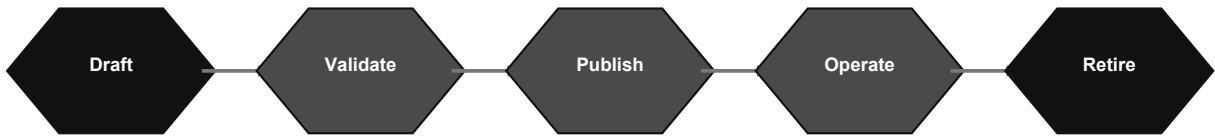
$$\text{share}_i = B_{\text{epoch}} \times e_i \div \sum_j (e_j), \text{ where } e_i = \min(u_i, \text{cap}_i) \times r_i \times a_i$$

Here B_{epoch} is a pre-funded epoch budget, u_i is accepted metered work, cap_i limits concentration, r_i is a published reliability factor, and a_i is an audit or acceptance factor. If total eligible work is zero, no distribution is made. This is a candidate accounting rule, not a token emission promise or guaranteed provider return.

4. Agent Lifecycle and Commerce

An agent package moves through draft, validation, publication, operation, and retirement. Publication requires a manifest, maintainer identity, version identifier, declared data practices, and a minimal safety and interface test. Updates create new versions rather than silently changing a published artifact. Consumers can pin a version or accept a specified upgrade policy.

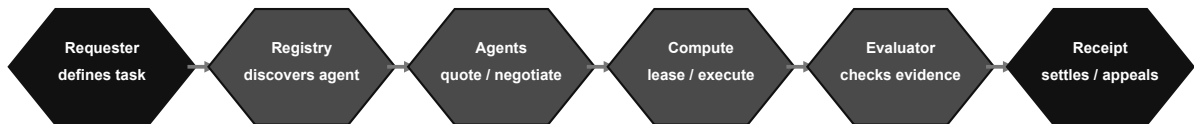
AGENT PACKAGE LIFECYCLE



Each release is versioned; rollback restores a prior signed manifest.

A task agreement records requester, provider, scope, price or budget ceiling, deadline, acceptance test, permitted data use, and dispute path. Completion produces a receipt linking task identifier, agent version, input commitment where appropriate, output commitment, timestamps, usage, and evaluator result. The receipt does not expose private payload. On-chain settlement is optional and should be used only where auditability and interoperability outweigh transaction cost and privacy risk.

AGENT-TO-AGENT SERVICE MARKET



Every claim states what is checked - and what remains unverified.

An agent may discover another agent, request a capability quote, lease compute capacity, or subcontract a bounded task. A lease should specify resource class, benchmark profile, reserved time window, price ceiling, data-handling policy, completion evidence, and cancellation terms. Capacity is not a promise of model quality. Nexora should settle actual measured usage against the accepted quote, with a dispute window and a bounded retry policy.

$$C_{\text{task}} = \sum_r (u_r \times p_r) + C_{\text{verification}} + C_{\text{network}}$$

$$\text{lease}_{\text{utilization}} = \min(1, \text{used}_{\text{units}} \div \text{reserved}_{\text{units}})$$

The cost equation is an accounting identity, not a pricing oracle. Unit definitions and metering sources must be agreed before execution; unused reserved capacity may be handled by a disclosed cancellation rule. For multi-agent work, the parent receipt references child receipts and the allocation rule so the same cost is not charged twice.

5. Quality, Reputation, and Incentives

Nexora separates task acceptance, service reliability, uncertainty handling, and policy compliance. Each component is domain-specific, normalized to [0,1], and disclosed with its evidence count.

$$q_k = w_a A_k + w_r R_k + w_c C_k + w_p P_k$$

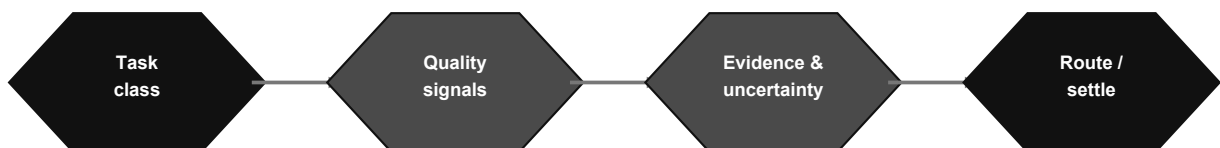
Symbol	Meaning	Evidence source
A_k	Task acceptance performance	Versioned tests or requester acceptance
R_k	Service reliability	Availability and successful completion records
C_k	Calibration and uncertainty handling	Calibration set and abstention behavior
P_k	Policy compliance	Audited permission and policy events
w_a, w_r, w_c, w_p	Published domain weights	Non-negative; sum to 1

A recency-weighted estimate is computed independently for each agent and task class. The half-life h is published; evidence older than h contributes half as much.

$$Q_i = \sum_k (v_k \times 2^{(-age_k \div h)} \times q_k) \div \sum_k (v_k \times 2^{(-age_k \div h)})$$

Parameter	Interpretation	Required display
v_k	Evaluation strength and capped task importance	Weighting rule and cap
age_k	Age of evidence in consistent time units	Evaluation timestamp
h	Evidence half-life	Value and rationale
n_i	Number of observations	Raw count by task class
$interval(Q_i)$	Estimate uncertainty	Credible or confidence interval

QUALITY → EVIDENCE → ROUTING



Acceptance ≠ reliability ≠ calibration ≠ policy compliance

A future incentive pool, if adopted, pays only from funds already deposited and only for eligible contributions.

$$x_i = \max(0, Q_i - q_{\min}) \times u_i \times d_i$$

$$\text{reward}_i = B_{\text{pool}} \times x_i \div \sum_j (x_j)$$

Variable	Definition	Control
B_{pool}	Funds already deposited in the pool	No unfunded emissions are implied
q_{min}	Minimum quality threshold	Published before the epoch
u_i	Verified use or accepted work	Rate limits and receipt checks
d_i	Diversity / concentration adjustment	Bounded; documented calculation
x_i	Eligible contribution weight	Non-negative; zero below threshold

If $\text{sum}_j(x_j)=0$, no payout is made and the pool follows a published carry-forward or refund rule. This is a design option, not a promised yield. Parameters, caps, funding source, and appeals must be approved before launch. A single score must never be the sole basis for access, payment, or penalties.

6. Coordination and Robust Evaluation

Nexora can use multiple independent evaluators to assess a work product against an agreed rubric. Let $e_{\{k,m\}}$ in $[0,1]$ be evaluator m 's score for task k , and let reliability weight h_m be estimated from blind calibration sets and audit history. A robust aggregate can be the weighted median: $q_k = \text{weighted_median}(e_{\{k,m\}}; h_m)$. A weighted median resists a limited number of extreme ratings better than a simple mean, but cannot protect against a coordinated majority or poor rubric.

For higher-stakes tasks, use hidden test cases, evaluator rotation, disclosed conflicts, reproducibility checks, and an appeal window. Track disagreement as uncertainty: $U_k = \text{weighted_MAD}(e_{\{k,m\}})$, where MAD is median absolute deviation. High disagreement triggers review rather than automatic settlement. Where objective tests do not exist, label the result as subjective assessment and expose the rubric.

7. Safety, Privacy, and Security

Agent identity is not authorization. Nexora should issue narrowly scoped capabilities that specify action, resource, spending limit, duration, and approval requirements. A high-impact action requires step-up approval by the user or an organization policy engine. Tool adapters should validate inputs, enforce quotas, and preserve an audit trail.

Threats include prompt injection, compromised dependencies, credential theft, evaluator collusion, fake usage, data exfiltration, and malicious package updates. Mitigations include signed manifests, dependency pinning, sandboxing, secret isolation, content and tool-call policies, rate limits, provenance records, revocable credentials, and incident response. These reduce risk but do not guarantee safety.

Private data should remain in the execution environment whenever feasible. A commitment $H(x)$ can bind a record to a cryptographic digest without publishing x , but it does not prove that x is accurate, lawful, or fairly collected. Zero-knowledge proofs or trusted execution may be explored for narrowly defined attestations, with their assumptions stated explicitly.

VERIFIABILITY CLAIM BOUNDARIES



Every claim states what is checked - and what remains unverified.

Verifiability is claim-specific. A commitment can show that a disclosed artifact matches a previously committed digest; a signature can bind an action to a key; a deterministic test can check a stated property; an execution attestation can report a

runtime measurement under hardware assumptions. None alone proves that an answer is correct, that training data was lawful, or that an opaque model followed the claimed reasoning. Nexora should label each receipt with the exact claim type and verification method.

$$V_{\text{claim}} = (\text{commitment}, \text{signer}, \text{test}_{\text{suite}}, \text{execution}_{\text{context}}, \text{verifier}_{\text{set}})$$

A field may be unknown; unknown must not be silently interpreted as verified. For model outputs, use layered evidence where practical: provenance and version binding, task-specific tests, independent evaluators, replayable tool traces, and a calibrated uncertainty statement. If a test is nondeterministic or relies on an oracle, record that dependency and its limitations.

8. Governance and Disputes

Governance should begin with clear roles: protocol maintainers, registry operators, evaluators, service providers, users, and an independent appeals function. Proposed changes require public specifications, compatibility notes, security review, and a staged rollout. Critical changes should include a delay and rollback path.

Disputes should distinguish non-delivery, specification mismatch, low-quality output, policy violation, and external outage. The process should preserve evidence, allow both parties to respond, and apply remedies proportionate to the agreement. Automated scoring may triage disputes but should not be the only authority for consequential cases. Governance participation does not transfer legal responsibility away from the entity that operates or uses an agent.

9. Economic Model

Nexora’s economic design links service execution, physical resource contribution, evidence production, and protocol operations. The architecture separates payment for delivered services from any optional incentive pool. A ledger can record commitments and settlement events; parties still define the service contract, accepted evidence, and remedies.

ECONOMIC FLOWS



Every claim states what is checked - and what remains unverified.

A task budget is decomposed before dispatch. The requester can approve the total or set component ceilings; the runtime rejects any quote that exceeds the authorized amount without renewed approval.

$$P_{\text{quote}} = P_{\text{model}} + P_{\text{compute}} + P_{\text{tools}} + P_{\text{storage}} + P_{\text{evaluation}} + F_{\text{protocol}}$$

$$P_{\text{paid}} \leq B_{\text{authorized}}$$

$$B_{\text{authorized}} = B_{\text{execution}} + B_{\text{evaluation}} + B_{\text{contingency}}$$

The accounting model reconciles quoted, metered, accepted, disputed, refunded, and settled amounts. Compute providers are paid against accepted meter records; agent operators are paid under the task agreement; evaluators are compensated under a disclosed evaluation contract. Any protocol fee is itemized before execution. Contingency is not revenue; unused amounts follow the task’s cancellation rule.

Economic circuit	Flow object	Settlement condition
Service market	Requester budget to agent/operator	Accepted deliverable and task receipt
Resource market	Compute/storage lease to physical provider	Metered units within the signed capacity and price ceiling
Evidence market	Evaluation budget to reviewer or test operator	Rubric version, result, and provenance attached
Protocol operations	Disclosed platform fee	Fee schedule and recipient set before authorization
Optional incentive pool	Pre-funded contribution reward	Eligibility, cap, and appeal rules published before epoch

Proposed token allocation (discussion parameters)

The proposed maximum supply is fixed at 10,000,000,000 NEXORA units. The table below is an illustrative allocation for discussion; it is not an approved issuance, sale, or return promise. Each bucket is a share of the fixed maximum supply, so the percentages sum to 100%.

Allocation bucket	Share	Units (10 billion max)	Proposed release principle
Community governance & ecosystem	35%	3,500,000,000	Distributed through published grants, participation programs, and governance-approved budgets over 8 years
Early users & contributors	15%	1,500,000,000	Eligibility tied to verifiable contribution; proposed 20% at launch and remainder linearly over 18 months
Core team	15%	1,500,000,000	12-month cliff, then monthly vesting over 36 months
Early backers	15%	1,500,000,000	6-month cliff, then monthly vesting over 24 months
Protocol treasury	10%	1,000,000,000	No automatic unlock; use by disclosed governance decision and public reporting
Liquidity & launch operations	5%	500,000,000	Budgeted for initial liquidity and launch costs; addresses and uses disclosed before deployment
Foundation, security & research	5%	500,000,000	12-month cliff, then monthly vesting over 36 months for protocol support and audits

$$T_{\text{bucket}} = 10,000,000,000 \times \text{allocation}_{\text{share}}$$

The proposed schedule does not imply that tokens are currently issued or that any category has a legal claim. Before adoption, the project should publish eligibility rules, wallet controls, cliff and vesting start dates, treatment of unallocated balances, and a method for reporting circulating and locked supply. Any change should apply prospectively and be documented through the governance process. Incentives must be funded from an explicit allocation or other disclosed source; they do not create a guaranteed yield.

10. Roadmap and Success Measures

IMPLEMENTATION GATES



Every claim states what is checked - and what remains unverified.

Gate	Deliverable	Exit criterion
0 Specification	Manifest, task receipt, permission schema	Independent parties validate the same test vectors
1 Runtime	Registry, SDK, workflow runner, adapters	A bounded multi-agent task completes with cost and permission controls
2 Evidence	Version-pinned terms, evaluator calibration, dispute flow	Acceptance and dispute outcomes are auditable
3 Open coordination	Multiple operators and portable records	Discovery, verification, settlement, and exit do not require one operator
4 Governance	Shared parameter process and migration controls	Change, rollback, and emergency paths pass operational review

Phase 0 - Specification: define agent manifest, task receipt, permissions, evaluation rubric, and reference threat model. Success: independent implementers can validate the same package and receipt.

Phase 1 - Usable network: publish a searchable registry, SDK, workflow runner, and provider adapters. Success: users can compose multi-agent workflows with clear cost and permissions.

Phase 2 - Evidence and marketplace: add version-pinned task agreements, receipts, evaluator calibration, and dispute workflows. Success: repeat usage, low unresolved dispute rates, and measurable service reliability.

Phase 3 - Open coordination: support multiple registry and evaluator operators, portable reputation records, and optional interoperable settlement. Success: no single operator is required for discovery, verification, or exit.

Phase 4 - Governance expansion: consider community control of shared parameters after operational evidence and security audits. Success metrics should include task completion, acceptance rate, p95 latency, cost per accepted task, repeat-user rate, provider concentration, evaluator disagreement, incident frequency, and appeal outcomes. Targets should be set only after baseline measurement.

11. Formal System Model and Notation

This section defines a minimal mathematical model for the proposed protocol. It is intended to make assumptions inspectable and implementation choices testable. It is not a claim that one formula can capture every domain or that all inputs are objectively measurable. Scores must be accompanied by provenance, sample counts, and uncertainty. Subscripts identify a task or component; they are indices, not arithmetic operators. The division symbol denotes division, while $\leq$ is an upper bound.

A task is represented as a tuple (id, requester, capability, constraints, budget, deadline, acceptance_rule, privacy_class). An agent is a versioned service with a manifest and a set of permissions. A workflow is a directed acyclic graph $G=(V,E)$ unless a bounded loop is explicitly declared. Every node v in V is a task step, and an edge (u,v) means v depends on u . Cycles without a termination rule are invalid.

$$q_k = w_a A_k + w_r R_k + w_c C_k + w_p P_k$$

This composite score is defined only when each component is measured on [0,1], the weights are non-negative, and $\sum(w)=1$. A_k is acceptance-test performance, R_k service reliability, C_k uncertainty handling, and P_k policy compliance. The domain publishes weights and tests before a task starts. When a component is unavailable, renormalizing the remaining weights may create misleading comparisons; the system should mark the score incomplete or apply a declared missing-data rule.

Symbol	Meaning	Constraint / interpretation
q_k	Observed quality for task k	0 to 1; never a truth guarantee
w_a, w_r, w_c, w_p	Published component weights	Non-negative; sum equals 1
A,R,C,P	Acceptance, reliability, calibration, policy	Defined per domain before evaluation
n_k	Evidence count for an agent and task class	Shown with the score
U_k	Uncertainty indicator	Higher values trigger review or cautious routing

A task outcome should not be collapsed to a quality number alone. Its evidence record also includes the test version, evaluator identity class, agent version, timestamp, and whether the result was contested. Scores are task-class specific; translation across classes requires a separately validated mapping.

12. Capability-Aware Routing

Routing chooses an eligible agent or sub-workflow for a task. Eligibility is evaluated before ranking: an agent must support the required capability, meet data residency and permission constraints, fit the time window, and remain within the budget. The router must never turn a high score into permission to exceed the requesters authority.

$$U(i|k) = a Q(i,k) + b \text{Rel}(i,k) - c \text{Cost}(i,k) - d \text{Lat}(i,k) - e \text{Risk}(i,k)$$

$$i^* = \arg \max U(i|k), \text{ for } i \text{ in } E(k)$$

$E(k)$ is the eligible set after hard constraints. Q is quality for the task class, Rel is observed availability, Cost is normalized expected cost, Lat is normalized latency, and Risk captures declared operational risk. Coefficients a-e are policy parameters selected by the requester or organization. Cost and latency normalization ranges must be published; otherwise changing units changes the ranking. If $E(k)$ is empty, the system should stop and ask for a revised constraint rather than silently relax a hard requirement.

For exploration among similarly ranked eligible providers, a temperature-controlled distribution may be used only when the user has opted into exploration: $p(i|k) = \exp(U(i|k)/T) / \sum_j \exp(U(j|k)/T)$. A low T concentrates on high-scoring options; a higher T increases exploration. The interface should show the selected provider, estimated cost range, expected latency, and reason for selection before any consequential external action.

13. Reputation With Uncertainty

An observed pass/fail result can update a beta-binomial reliability estimate. For a given task class, initialize α_0 and β_0 as declared prior parameters. Let $y_k=1$ for a successful acceptance test and 0 otherwise.

$$\alpha_n = \alpha_0 + \sum_k y_k$$

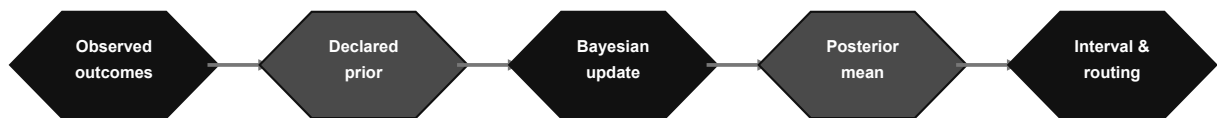
$$\beta_n = \beta_0 + n - \sum_k y_k$$

$$E[\theta|\text{data}] = \alpha_n \div (\alpha_n + \beta_n)$$

The posterior mean estimates the probability of passing comparable tests under the model assumptions. The credible interval must be displayed; small samples produce wide intervals. Priors are assumptions and should be visible. Results from materially different task classes must not be pooled without evidence that the populations are comparable. Time decay may be applied to stale observations, but the decay rule must be declared and raw counts retained.

Worked example	Value	Interpretation
Prior	Beta(1,1)	Uniform starting assumption for illustration
Observations	8 passes; 2 failures	One task class and one test version
Posterior	Beta(9,3)	Prior parameters plus observed outcomes
Posterior mean	9 / (9 + 3) = 0.75	Estimate only; interval still matters

EVIDENCE TO POSTERIOR ESTIMATE



Sparse evidence widens the interval; it should not be displayed as certainty.

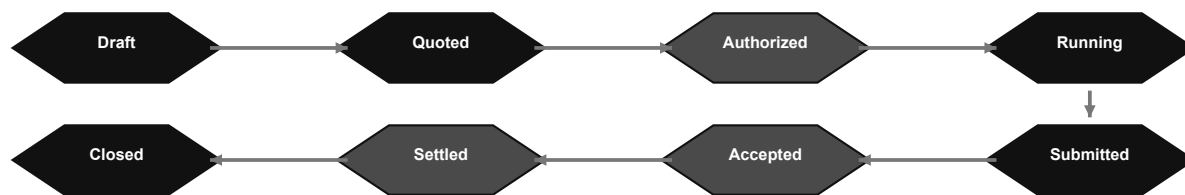
Display field	Why it matters
Posterior mean	Compact estimate of observed pass probability
Credible interval	Shows uncertainty and guards against false precision
Sample count	Lets users distinguish mature from sparse evidence
Task class and test version	Defines where the estimate may generalize
Last observation date	Signals possible model or service drift
Dispute / audit flag	Prevents contested evidence from appearing settled

Evaluator disagreement is a review signal, not proof of low quality.

14. Workflow State Machine and Task Receipt

Every delegated task moves through an explicit state machine. State transitions are append-only; corrections add a new event rather than rewriting prior evidence. The runtime prevents an agent from claiming completion before outputs and required evidence are attached.

TASK STATE TRANSITIONS



Dispute / reject branches re-enter after review.

State	Entry condition	Allowed next states
Draft	Requester is defining scope	Quoted, Cancelled
Quoted	Provider, price, terms and tests are known	Authorized, Rejected, Expired
Authorized	Required permissions and budget approval exist	Running, Cancelled
Running	Execution has started within scope	Submitted, Failed, Cancelled
Submitted	Output and receipt are attached	Accepted, Disputed
Accepted	Acceptance rule passed or requester confirms	Settled, Closed
Disputed	A party contests evidence or terms	Accepted, Rejected, Escalated
Settled	Payment or other agreed settlement completed	Closed

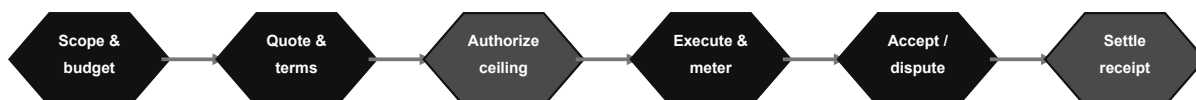
The receipt commits to task ID, manifest digest, package version, policy version, usage meter, output digest, timestamps, evaluator record, and settlement state. It may reference encrypted or access-controlled evidence off-chain. A digest establishes integrity relative to committed bytes, not truth of content or identity of the supplier.

Record group	Fields	Handling principle
Task identity	task_id; agent_version	Versioned and linkable
Scope & authority	policy_version; budget_cap	Keep private scope data access-controlled
Input/output binding	H(input); H(output)	Digest binds bytes; does not prove truth
Usage meter	model; tools; storage	Reconcile against quote and cap
Evaluation	rubric_version; result; interval	Append corrections; retain provenance
Settlement	state; amount; timestamp	Auditable and appealable

15. Settlement and Budget Invariants

The requester authorizes a maximum budget B before execution. A provider quote may include model usage, external tool usage, storage, and a platform fee. Settlement occurs only after the agreed acceptance condition or a resolved dispute.

BUDGET AND SETTLEMENT FLOW



Invariant: paid amount never exceeds newly authorized budget.

$$P_{\text{quote}} = p_{\text{model}} + p_{\text{tools}} + p_{\text{storage}} + p_{\text{service}}$$

$$P_{\text{paid}} \leq B_{\text{authorized}}$$

$$P_{\text{paid}} = I(\text{accepted}) \times P_{\text{quote}} + P_{\text{resolved}}$$

P_{resolved} applies only to disputed work and is constrained to the authorized budget. The equation is schematic: partial acceptance, cancellation fees, refunds, taxes, and third-party charges must be defined in the task agreement. No runtime action may increase $B_{\text{authorized}}$ without new approval.

16. Abuse Resistance and Evaluator Integrity

No scoring design removes collusion risk. Nexora should use defense in depth: evaluator rotation, blind calibration tasks, conflict disclosure, rate limits, identity or stake friction proportionate to the risk, anomaly detection, and appeal. Stake or identity checks are not a substitute for evidence and can create exclusion costs.

For evaluator m , let z_m be agreement with adjudicated calibration examples and let n_m be the number of such examples. A smoothed reliability estimate can be recorded as $h_m = (a_0 + z_m) / (a_0 + b_0 + n_m)$, where a_0, b_0 are declared smoothing parameters. Use h_m as one input to evaluator selection, not as an unquestioned trust score. Calibration results should be scoped to rubric and domain.

For repeated ratings by a suspected cluster, compare timing, shared funding, graph proximity, and unusually correlated score residuals. Such signals can trigger review, but correlation alone is not proof of misconduct. Remedies should be reversible where possible and include notice and appeal. Any automated penalty must be bounded by the agreement and the governing rules.

17. Threat Model and Control Matrix

Security controls should map to assets, threats, and accountable owners. A control is meaningful only if it is implemented, monitored, and tested.

Asset / threat	Preventive control	Detection and recovery
User credentials stolen	Scoped short-lived credentials; secret isolation	Revoke keys; incident notice; audit access
Malicious agent package	Signed manifest; pinned dependencies; sandbox	Behavior monitoring; quarantine version; rollback
Prompt injection via tool output	Treat retrieved content as untrusted; tool allowlist	Log calls; block policy violations; human review
Private data leakage	Data minimization; egress policy; redaction	DLP alerts; revoke access; breach response
Fake usage or Sybil ratings	Metered receipts; rate and identity friction	Anomaly review; re-score; appeal
Evaluator collusion	Blind tests; rotation; diverse operators	Disagreement analysis; independent adjudication
Ledger or provider outage	Idempotent task IDs; retry limits; fallback policy	Reconcile receipts; pause settlement; status page

For each threat, assign a risk owner and an operational response time. A public ledger can improve tamper evidence but may worsen data permanence and metadata exposure. Security review should consider the full system, including wallets, endpoints, package registries, and support processes.

18. Performance and Capacity Model

Capacity planning should be based on measured workload, not token or agent counts. For a workflow DAG, critical path latency is the longest dependency path, while total compute cost is the sum of metered costs across executed nodes. Parallel steps reduce wall-clock time only when resources and external services permit.

$$L_{\text{workflow}} \geq \max_{p \text{ in paths}(G)} \sum_{v \text{ in } p} L_v$$
$$C_{\text{workflow}} = \sum_{v \text{ in } V(\text{executed})} (C_{\text{model},v} + C_{\text{tool},v} + C_{\text{storage},v})$$

The first expression is a lower bound on latency; queuing, retries, and human approvals add delay. Report median and p95 latency separately, and segment by task class and provider. Retries should be bounded and idempotent to avoid duplicate charges or actions. Caching is permitted only when data policy and freshness requirements allow it.

Operational dashboards should expose request volume, completion rate, timeout rate, cost per accepted task, model/provider concentration, evaluator coverage, unresolved disputes, and security incidents. Each metric needs a definition, window, denominator, owner, and data-quality check.

19. Governance Parameters and Change Control

Shared parameters include manifest versions, supported signature schemes, evaluator policy, and settlement rules. A parameter change proposal should state the motivation, affected actors, migration path, security implications, and measurable rollback trigger.

Change class	Review requirement	Activation approach
Documentation / non-breaking	Maintainer review and public notice	Immediate after publication
Runtime behavior / compatible	Test vectors and security review	Canary deployment, then staged rollout
Settlement or permission semantics	User and provider impact analysis; explicit consent	Versioned task agreements; delayed activation
Key or ledger migration	Independent audit and recovery rehearsal	Opt-in migration with rollback window
Emergency containment	Named incident authority and recorded rationale	Immediate, time-limited, post-incident review

A governance process should publish participation thresholds and quorum rules before a vote. Token-weighted voting, if ever used, must disclose concentration, delegation, and capture risks. The protocol should maintain an exit path: users can export task receipts and agent manifests in documented formats even if an operator changes or service is discontinued.

20. Implementation Sequence and Validation Gates

The roadmap is a sequence of evidence gates rather than calendar promises. Gate A: interoperable manifest and signature verification. Gate B: bounded execution with permission enforcement and cost metering. Gate C: task receipts and a reproducible acceptance harness. Gate D: multi-evaluator assessment with calibration and appeals. Gate E: settlement interoperability and multi-operator registry. A gate is passed only when an independent implementation can reproduce the result against published test vectors.

Before expanding incentives, measure organic repeat use, quality by domain, evaluator disagreement, disputes per accepted task, concentration, and incident rate. Before expanding decentralization, demonstrate that participants can discover, verify, settle, and exit without a single operator. If these conditions are not met, remain at the simpler operational stage. This keeps protocol complexity proportional to demonstrated need.

Conclusion

Nexora proposes an accountable coordination layer for an AI ecosystem: agents that can be discovered and composed, tasks with explicit terms, actions bounded by permissions, and evidence that supports evaluation and settlement. The strongest foundation is not a speculative token or a claim of perfect decentralization. It is a set of useful interfaces and verifiable operating practices that let users trust specific work for specific reasons. The system should earn the right to decentralize as real participants, measurable utility, and credible safeguards emerge.