OOBE Protocol: A new execution architecture for autonomous systems v0.1

Alex Kelly

k6@oobeprotocol.ai

Abstract

Autonomous software systems are becoming increasingly capable, yet they remain constrained by unreliable infrastructure, opaque execution, and poorly aligned economic models. While blockchains provide cryptographic trust and settlement, they are not designed for high-frequency, continuous execution required by autonomous agents. Conversely, existing AI and agent frameworks lack verifiability, accountability, and native economic coordination. OOBE Protocol introduces a unified execution substrate for autonomous systems. It combines a deterministic agent runtime, an AI-tailored execution fabric (Synapse), and cryptographic settlement on Solana into a single coherent architecture. OOBE enables autonomous agents to plan, execute, verify, and pay for computation and infrastructure access in a reliable, observable, and economically enforceable manner.

1. Introduction

Autonomous agents already operate across finance, infrastructure, and data systems. However, they rely on brittle execution paths, centralized APIs, and opaque billing. These constraints prevent agents from operating safely and continuously at scale, OOBE Protocol addresses this mismatch by unifying agent execution, infrastructure access, and economic enforcement under a single runtime model.

2. Problem Statement

2.1 Execution Fragility

Agents depend on third-party RPC providers and APIs that fail unpredictably, throttle arbitrarily, and provide no execution guarantees.

2.2 Lack of Determinism

Agent behaviour is difficult to inspect, reproduce, audit, and debug. Making production deployment risky.

2.3 No Native Economic Enforcement

There is no protocol-level mechanism for usage-based billing, prioritization, or verifiable receipts.

2.4 Blockchain–Execution Mismatch

Blockchains provide trust and settlement, but are inefficient for continuous execution, streaming, retries, and adaptive workflows.

3. Design Principles

1. **Execution First**
   Autonomous systems are execution systems.
2. **Determinism and Observability**
   Executions must be inspectable and reconstructible.
3. **Economic Enforcement**
   Access must be priced, prioritized, and verifiable.
4. **Trust Minimization**
   Agents are not trusted by default.
5. **Layered Responsibility**
   Planning, execution, and settlement are separated but integrated.

This follows established end-to-end system design principles [6]. The separation of planning, execution, and settlement follows modular system design principles applied in modern blockchain architectures [5].

3.1 Non-Goals

OOBE Protocol is not designed to:

- Replace blockchain consensus mechanisms
- Perform general-purpose on-chain computation beyond settlement, verification and proof validation.

- Provide guaranteed economic outcomes or yields
- Act as an autonomous governance authority

The protocol intentionally focuses on execution reliability, economic enforcement, and agent coordination rather than broad generalisation.

4. System Overview

OOBE Protocol consists of three tightly coupled layers:

- **OOBE Agent Runtime**

  Planning, orchestration, state, lifecycle control

- **Synapse Execution Fabric**

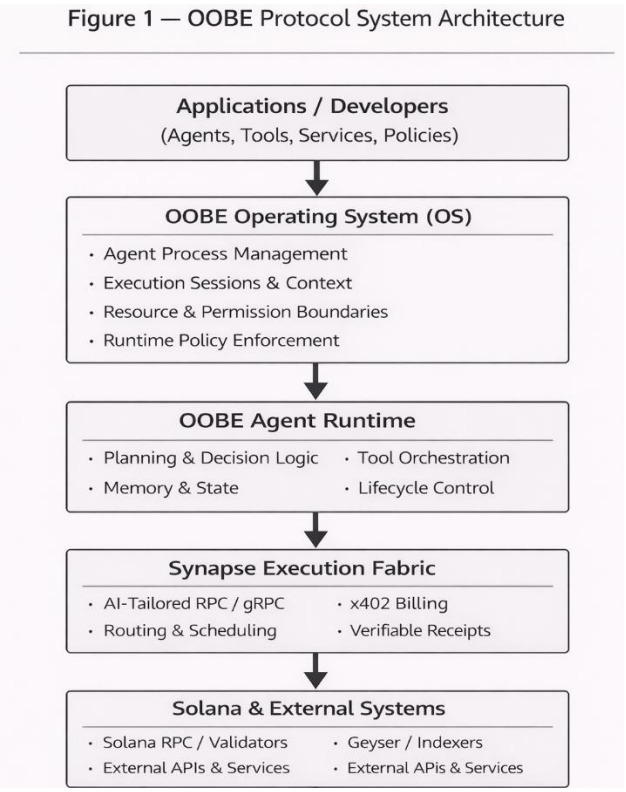
  Routing, scheduling, billing, verification

- **Solana Settlement Layer**

  Payments, receipts, cryptographic trust

Agents never interact directly with blockchains or infrastructure; all execution is mediated by Synapse. This layered separation mirrors modular blockchain architectures [1][5]. As shown in Figure 1.

Figure 1 — OOBE Protocol System Architecture



OOBE Protocol separates agent planning, execution enforcement, and cryptographic settlement into distinct but tightly integrated layers. All execution is mediated by Synapse.

5. High-Level Architecture

5.1 Application Layer

Developers build agents, tools, services, and policies. Applications express **intent**, not infrastructure details. This layered separation mirrors modular blockchain architectures [1][5].

5.2 OOBE Operating System (OS)

The OOBE Operating System (OS) provides the control plane for agent execution. It is responsible for instantiating agents, enforcing isolation, and applying execution policies before any agent logic is evaluated.

The OS manages execution sessions, capability boundaries, configuration loading, and policy enforcement. It defines the permissible execution environment in which agents may operate, independent of agent reasoning or application logic. This layer establishes deterministic execution constraints and safety guarantees between developer-defined intent and runtime execution.

5.3 Agent Runtime Layer (OOBE)

The OOBE Agent Runtime implements agent-level reasoning and execution within the environment defined by the OOBE Operating System.

It is responsible for deterministic planning, tool invocation, state progression, and execution flow. The runtime operates entirely within OS-defined constraints and does not manage permissions, resources, or policy enforcement. This separation allows agent behaviour to remain reproducible and inspectable while delegating execution safety and economic enforcement to lower layers.

5.4 Execution Fabric Layer (Synapse)

A **policy-enforcing execution fabric** providing:

- AI-tailored RPC / gRPC
- Priority routing and scheduling
- Streaming, batching, caching
- x402-based billing
- Cryptographically verifiable receipts

The execution fabric is designed to enforce execution guarantees without embedding application-specific logic [6].

5.5 Settlement & External Systems

Solana validators, RPC infrastructure, Geyser plugins, indexers, and external services execute state transitions and provide cryptographic settlement and finality for protocol operations.

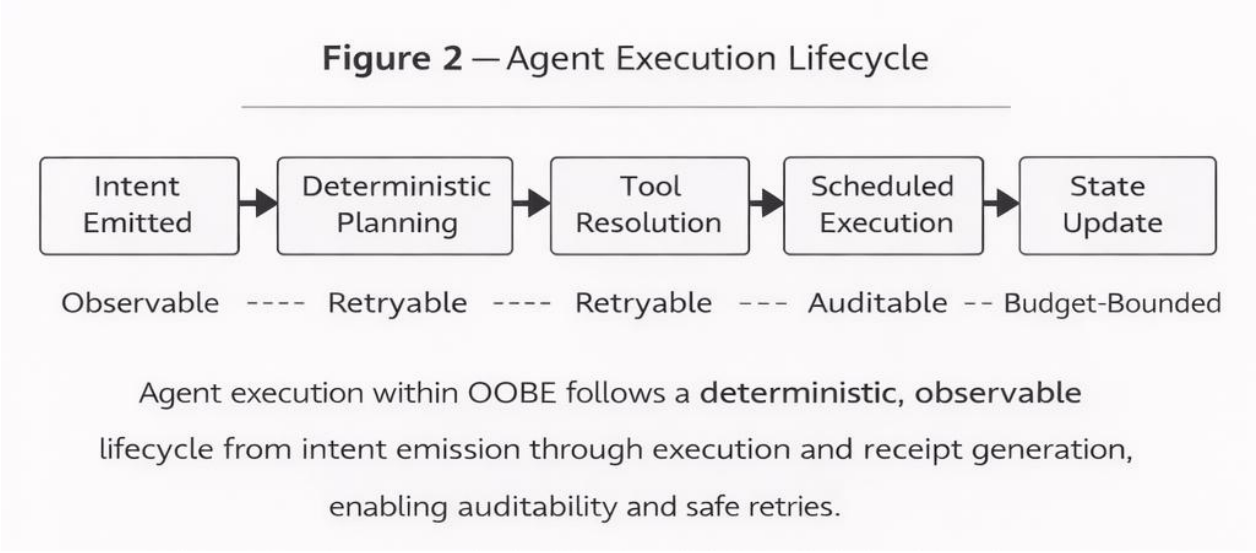
6. Agent Execution Lifecycle

1. Intent emission

2. Deterministic planning
3. Tool resolution
4. Scheduled execution
5. Streaming of partial results
6. State update
7. Final output with receipts

Properties: deterministic, observable, streamable, retryable, auditable.

As shown in Figure 2.



7. AI-Tailored RPC Design

Synapse RPC is designed for autonomous systems:

- Session-aware execution
- Continuous loops
- Parallel and pipelined calls
- Streaming results
- Native micro-billing via x402
- Built-in QoS and prioritization
- Verifiable usage receipts

Agents are first-class computational actors. Payment-gated execution is aligned with HTTP-native payment semantics originally proposed via the 402 “Payment Required” status code [3].

8. Agent–RPC Interaction Model

Agents submit structured execution intents. Synapse:

- Queues and prioritizes
- Enforces rate limits and cost ceilings

- Executes against chains and services
- Streams results and returns receipts

Agents are bounded and isolated by default.

9. Solana Integration

OOBE leverages Solana's architecture for high-throughput settlement and cryptographic finality, consistent with its performance-first design principles [1]:

- Parallel transaction execution
- Account-based conflict detection
- Low-latency settlement
- Memo v2 anchoring
- PDA-based receipts
- Account state indexing
- Real-time indexing via Geyser

Solana anchors truth without acting as the execution engine.

10. Security Model

Assumptions: agents may be buggy or malicious; infrastructure and networks may fail; Solana may stall or fork.

Mitigations: isolation, rate limiting, retries and rerouting, payment-based suspension, state reconstruction, graceful degradation.

Failure assumptions reflect standard reliability engineering practices [9]. Horizontal scaling and resource allocation strategies reflect proven approaches from large-scale distributed systems [7].

11. Scalability Model

Bottlenecks: RPC throughput, disk I/O, agent memory growth, latency, bandwidth.

Strategies: horizontal RPC scaling, sharded indexing, tiered storage, streaming over polling, local caching in Synapse.

Optimized for high concurrency.

12. On-Chain vs Off-Chain Responsibilities

On-chain responsibilities include payments, execution receipts, trust anchors, cryptographic proofs, and protocol-level state required for verifiable execution and settlement.

On-chain computation is limited to state verification, proof validation, and execution integrity checks, rather than general-purpose or agent-level computation.

Off-chain responsibilities include agent planning, optimization, orchestration, and auxiliary computation where performance or flexibility is prioritized over finality.

Truth and critical execution state are anchored on-chain. Performance-sensitive reasoning and coordination occur off-chain.

On-chain anchoring of receipts follows established principles of deterministic state verification [2].

13. Developer Primitives

- Agents – agent behaviour, state transitions, and decision logic
- Tools - executable capabilities
- Receipts - verifiable execution and payment records
- Developers build systems, not infrastructure.

14. Economic Model

OOBE Protocol operates under an execution-driven economic model. Autonomous agents consume execution, bandwidth, and reliability through Synapse, and pay for these resources on a usage basis.

14.1 Payment Flow

1. An agent initiates an execution session via Synapse
2. Execution is priced based on throughput, priority, duration, and resource intensity
3. Payments are enforced via x402-compatible mechanisms
4. Each execution produces a cryptographically verifiable receipt
5. Execution halts automatically upon payment failure or budget exhaustion

Payments are enforced prior to execution, reflecting web-native payment-before-service semantics [3]

14.2 Pricing and Prioritization

Execution access is tiered based on:

- Latency guarantees
- Throughput requirements
- Execution frequency
- Session duration

Higher tiers receive preferential routing and scheduling.

14.2.1 Monetization Modes (Usage and Subscriptions)

Synapse may support multiple monetization modes depending on workload characteristics. Pay-per-use is enforced per execution session via x402-compatible flows, while subscription tiers may provide predefined capacity envelopes, priority routing, and service-level features. Subscription access does not remove usage enforcement; it scopes pricing, limits, and quality-of-service parameters for eligible sessions.

14.3 Infrastructure Incentives

Infrastructure providers are entities that supply execution capacity, networking, and system resources to the OOBE Protocol, including operators of Synapse execution nodes, RPC endpoints, indexing services, and other execution-critical infrastructure required for agent operation.

Participation is economically incentivized through usage-based compensation tied directly to execution quality and reliability, rather than nominal availability. Incentives are designed to reward sustained performance under real workloads.

Infrastructure providers are incentivized to:

- Maintain reliable execution under sustained and burst load
- Compete on latency, availability, and execution consistency
- Deliver cryptographically verifiable execution results and receipts

Performance metrics such as latency, error rates, execution completeness, and receipt validity are continuously observed and evaluated by the Synapse execution fabric. During early phases, infrastructure participation may be permissioned or selectively onboarded to ensure execution reliability and security. As the protocol matures, participation and routing policies may expand in accordance with the roadmap principle of decentralization following demonstrated reliability.

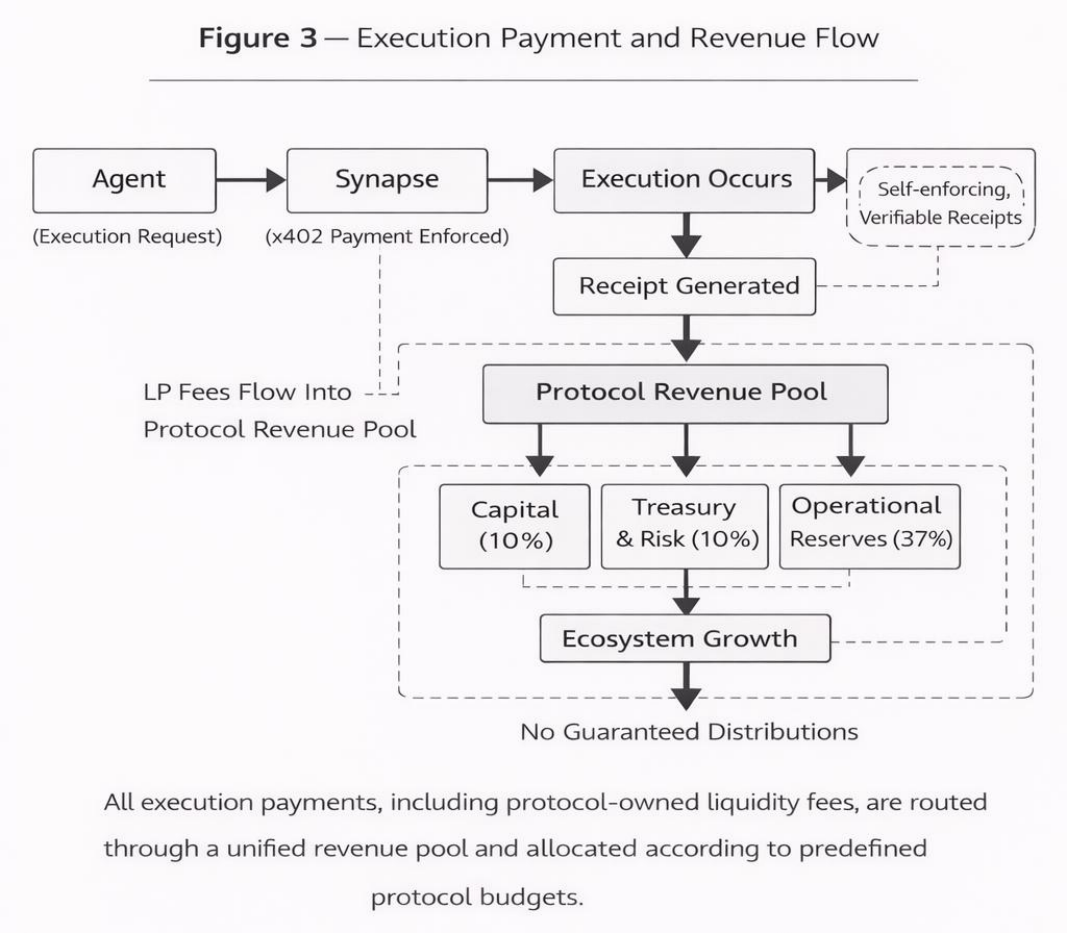
Providers that fail to meet performance thresholds are automatically deprioritized or excluded from routing. This establishes a competitive, performance-weighted infrastructure market governed by objective execution metrics rather than discretionary selection.

14.4 Revenue Sources

Protocol revenue **may** originate from:

- Execution and RPC usage fees
- Priority routing and session access
- Protocol-owned liquidity pool fees
- Enterprise or service-tier integrations, including Synapse subscription tiers and custom execution agreements

All revenue is treated uniformly at the protocol level.
As shown in Figure 3.



14.5 Revenue Allocation

Protocol revenue is allocated according to predefined operational and strategic budgets to ensure sustainability, growth, and infrastructure reliability.

Category	Allocation
Capital	10.0%
Treasury & Risk Buffer	10.0%
Operational Reserves	37.0%
Ecosystem Growth	18.0%
Core Infrastructure & R&D	25.0%
Total	100.0%

14.5.1 Operational Reserves (37.0%)

Operational reserves are primarily allocated to sustaining core team operations, with the majority directed toward team compensation, alongside a smaller allocation reserved for legal, compliance, and regulatory obligations.

Sub-Category	Allocation
Team Salaries	34.5%
Legal & Compliance	2.5%

14.5.2 Ecosystem Growth (18.0%)

Ecosystem Growth funds **may** be used to support adoption, participation, and long-term ecosystem development across OOBE:

- Community incentives intended to encourage engagement and usage
- Referral programs designed to support organic ecosystem growth
- Staking and participation mechanisms aligned with protocol coordination
- Discretionary token buybacks, subject to governance decisions and market conditions
- Developer grants, bounties, and tooling to support integrations and application development

Use of Ecosystem Growth funds is discretionary and does not constitute a guarantee of returns, incentives, or future outcomes.

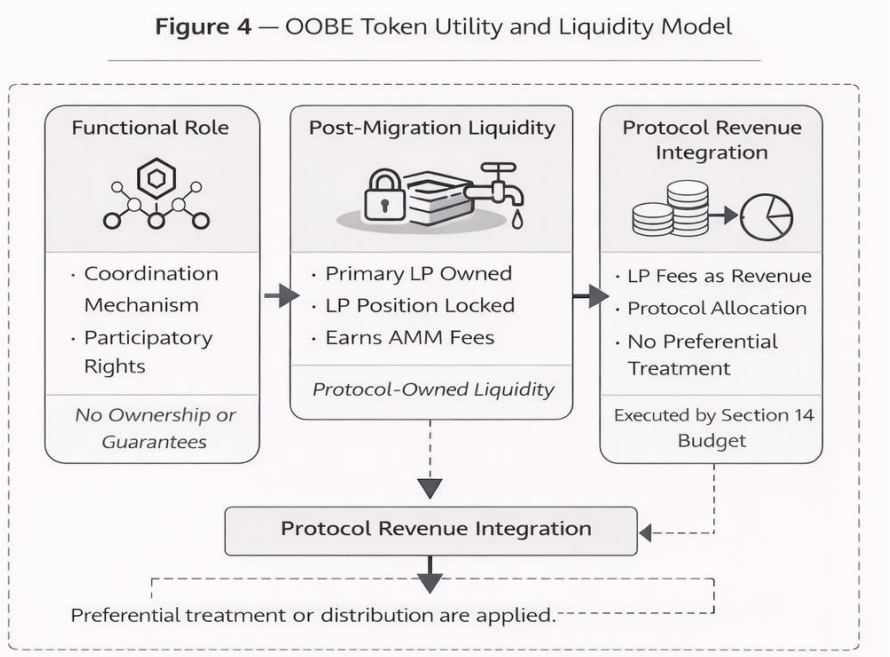
15. Token Design

15.1 Functional Role

The OOBE token functions as an economic coordination and participation mechanism within the protocol. It does not represent ownership, profit rights, or guaranteed value. The token's primary function is to coordinate access, prioritization, and participation within the execution economy, rather than to serve as a passive value-accrual or yield instrument.

Protocol revenue supports operations and ecosystem growth, while token utility governs access, coordination, and participation within the execution economy. These are intentionally separated to avoid conflating protocol sustainability with promises of holder returns. As shown in Figure 4.

Figure 4 — OOBE Token Utility and Liquidity Model



The OOBE token coordinates activity and migrates into a protocol-**owned liquidity** pool earning AMM-based fees, with **staking mechanisms** used for access and optional economic guarantees.

15.2 Post-Migration Liquidity Model

Following migration, OOBE Protocol will operate with protocol-owned liquidity as a core infrastructure component:

- The protocol will own its primary liquidity pool on Raydium
- The LP position will be locked to ensure long-term stability and alignment
- Standard AMM trading fees generated by the liquidity pool will accrue to the protocol
- Token utility will be directly linked to protocol usage, including access to Synapse execution tiers, subscription-based services, and differentiated execution capabilities

Protocol-owned liquidity is managed as infrastructure, supporting execution reliability, economic enforcement, and long-term sustainability. It is not designed or represented as a yield-bearing product.

15.3 LP Fee Integration

Liquidity pool fees are treated as protocol revenue and are allocated using the same revenue allocation model defined in Section 14.

The protocol **may** apply a **swap fee** within a defined range of **2–4%** on protocol-owned liquidity pools. These fees are intended to support protocol operations, infrastructure costs, and long-term sustainability, and are **not** designed as a yield mechanism or incentive program.

Swap fee ranges are intended to be adjustable and risk-aware, balancing operational sustainability with market competitiveness and liquidity depth.

Fee parameters are subject to governance oversight and may be adjusted over time in response to market conditions, liquidity depth, and protocol usage. Swap fees are applied at the liquidity pool level and are independent of agent execution fees charged through Synapse.

No preferential treatment or distribution is applied.

15.4 Staking and Participation

Staking mechanisms, if implemented, are intended to align long-term participants with the operational health and economic activity of the protocol.

Staking **may** be used to:

• Enable access to priority execution tiers and Synapse subscription features

• Provide economic guarantees or collateral for agent execution

• Participate in governance under TGAR constraints

• Receive variable incentives tied to protocol usage and participation

Staking does not imply fixed yields, guaranteed rewards, or direct entitlement to protocol revenue. Any incentives associated with staking are variable, discretionary, and dependent on protocol usage, performance, and governance decisions.

16. Governance Model

OOBE Protocol adopts a **Threshold-Governed Agent Runtime (TGAR)** governance framework. TGAR governance constraints are informed by prior DAO governance failures [8].

16.1 TGAR Overview

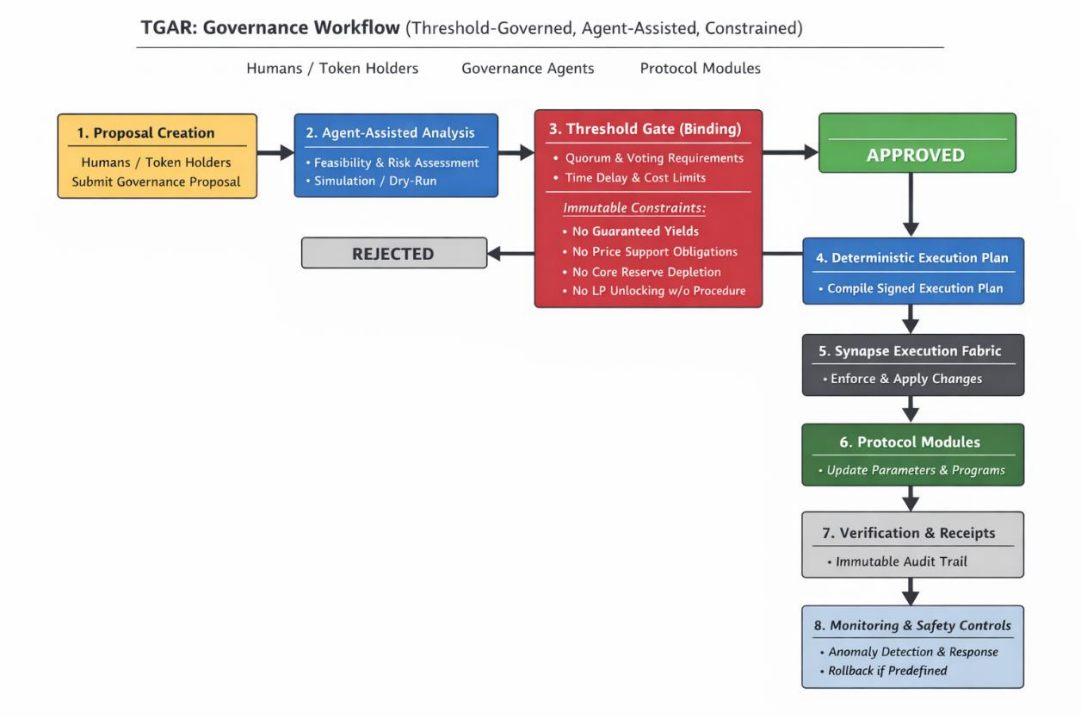
TGAR combines human decision-making with agent-assisted analysis and execution, enforced through protocol-level thresholds and immutable constraints. TGAR governance constraints reflect lessons from economic security and incentive alignment in shared infrastructure protocols [4]

Governance is treated as an execution workflow, not a discretionary authority.

16.2 Governance Flow

1. Humans or token holders submit proposals
2. Agents simulate feasibility, risk, and execution impact
3. Binding thresholds are evaluated
4. Approved actions are executed deterministically via Synapse
5. Receipts and audit trails are produced

The Threshold-Governed Agent Runtime (TGAR) governance flow (Figure 5) illustrates how governance proposals are submitted by humans, analysed and simulated by agents, admitted through binding thresholds and immutable constraints, and executed deterministically via the Synapse execution fabric, producing a verifiable audit trail.



16.2.1 Initial Governance Thresholds (v0.1)

At launch (v0.1), TGAR operates under conservative default thresholds designed to prioritise execution safety, economic discipline, and operational stability.

Initial governance thresholds include:

- **Proposal Submission:**
  Restricted to predefined governance actors and multisig-controlled entities during early phases.
- **Quorum Requirement:**
  A supermajority quorum is required for any governance action that affects economic parameters, infrastructure configuration, or liquidity management.
- **Voting Thresholds:**
  - Simple parameter adjustments require a qualified majority
  - Economic or treasury-affecting actions require a higher supermajority
- **Time Delays (Timelocks):**
  Approved proposals are subject to mandatory execution delays to allow for review, monitoring, and cancellation if anomalies are detected.
- **Budget and Impact Caps:**
  Governance actions are constrained by predefined cost ceilings and impact limits enforced at execution time.
- **Execution Safety Checks:**
  Agent-mediated simulations must pass execution feasibility, reliability, and constraint validation before actions are eligible for execution.

These thresholds may be adjusted over time through governance processes operating within the constraints defined in Section 16.3. Threshold relaxation is not assumed and is contingent on demonstrated protocol maturity and reliability.

16.3 Governance Constraints

Governance **cannot**:

- Commit to guaranteed yields or returns
- Promise price support or buyback schedules
- Deplete operational or infrastructure reserves below safety thresholds
- Disable paid-execution enforcement
- Unlock protocol-owned liquidity without predefined procedures

16.4 Governance Scope

Governance **may**:

- Adjust ecosystem incentives
- Tune protocol parameters within bounds
- Authorize discretionary buybacks
- Propose upgrades subject to constraints

TGAR ensures governance cannot undermine execution reliability.

17. Use Cases

OOBE Protocol is designed as a general-purpose execution substrate for autonomous systems. The following use cases illustrate how agent-based systems can leverage deterministic execution, economic enforcement, and verifiable infrastructure access through OOBE.

These examples are non-exhaustive and intended to demonstrate classes of applications rather than fixed products.

17.1 Autonomous Trading and Execution

Autonomous trading agents require:

- Continuous execution
- Low-latency infrastructure access
- Deterministic behaviour
- Verifiable execution history
- Enforced cost and risk limits

Using OOBE Protocol, trading agents can:

- Execute strategies across decentralized exchanges and on-chain venues
- Pay for RPC access, priority routing, and execution throughput programmatically
- Operate within predefined budgets and execution constraints
- Produce cryptographically verifiable receipts for every action

This enables trading systems that are auditable, bounded, and suitable for long-running autonomous operation.

17.2 Infrastructure Management and Automation

Infrastructure systems increasingly rely on automated agents to manage:

- Node operations
- Monitoring and alerting
- Failover and recovery
- Resource allocation

OOBE enables infrastructure agents to:

- Continuously observe system state
- Execute corrective actions deterministically
- Interact with blockchains, RPC endpoints, and off-chain services
- Pay for execution and data access without manual intervention

This allows infrastructure automation to operate reliably under economic and operational constraints.

17.3 AI Service Orchestration

Modern AI applications often require orchestration across multiple services, including:

- Model inference
- Data retrieval
- Indexing and storage
- External APIs

Through OOBE:

- Agents can coordinate multi-step AI workflows
- Execution can be streamed, retried, and verified
- Costs are enforced at the execution layer
- Service interactions are auditable and reproducible

This supports AI systems that operate continuously and predictably in production environments.

17.4 DAO Operations and Governance Execution

DAOs require execution mechanisms that are:

- Transparent
- Deterministic
- Resistant to misexecution
- Economically accountable

OOBE supports DAO operations by enabling:

- Agent-assisted proposal analysis and simulation
- Deterministic execution of approved governance actions
- Enforcement of governance constraints via TGAR
- Verifiable execution receipts anchored on-chain

This reduces reliance on manual multisig execution and opaque operational processes.

17.5 Cross-Agent Economies

As autonomous agents interact with one another, new economic structures emerge:

- Agents paying other agents for services
- Agents coordinating workloads
- Agents competing for execution priority

OOBE provides the substrate for such economies by:

- Enabling agents to transact programmatically
- Enforcing payment before execution
- Recording verifiable interaction receipts
- Supporting priority and quality-of-service differentiation

This allows autonomous systems to form bounded, auditable, and economically coherent agent networks.

17.6 General Applicability

These use cases share common requirements:

- Deterministic execution
- Observable behaviour
- Economic enforcement
- Infrastructure abstraction

OOBE Protocol is designed to satisfy these requirements across domains without being tailored to any single application category.

18. Roadmap

Phase 1 - Foundation & Core Execution (Q1 2026)

- x402 Synapse Explorer (Beta)
- Agent Operating System (OS)
- Company establishment, funding & grants
- Ecosystem partnerships
- Holders rewards integration
- x402 Agent Merchant Token
- Synapse referral program

Objective: ship real execution with real payments and observability.

Phase 2 - Agent Services & Activation (Q2 2026)

- Agent marketplace
- Mobile apps (Seekr iOS/Android)
- CEX listing applications
- Cross-platform runtime
- Yield farming & staking
- Coordinated marketing

Objective: discovery, usage, and sustainability.

Phase 3 - Agent Economy & Distributed Execution (Q3 2026)

- x402 agent economy
- On-chain vector memory (hybrid)
- Distributed runtimes + sub chain sync
- TGAR governance
- Global Synapse expansion (US & Asia)

Objective: distributed, coordinated agent systems.

Phase 4 - Ecosystem Maturity (Q4 2026)

- Enterprise integrations
- Ecosystem expansion
- Grants & bounties
- Cross-chain integrations
- Tier-1 CEX listings

Objective: global adoption and decentralization.

Roadmap Principles: execution before abstraction; economics before governance; decentralization after reliability.

19. Risks and Limitations

Economic mispricing, infrastructure concentration, agent misuse, regulatory uncertainty, and emergent behaviours are acknowledged risks. OOBE mitigates but does not eliminate them. Residual risks reflect known challenges in distributed systems and autonomous software operation [9].

20. Conclusion

OOBE Protocol is not an application.
It is an **execution substrate for autonomous systems** - an agent OS, a programmable execution fabric, and a paid, verifiable RPC network unified under a single execution model.

21. References

[1] Solana Labs. *Solana: A New Architecture for a High Performance Blockchain*.
https://solana.com/solana-whitepaper.pdf

[2] Ethereum Foundation. *Ethereum Yellow Paper*.
https://ethereum.github.io/yellowpaper/paper.pdf

[3] Mozilla Developer Network. *HTTP 402 – Payment Required*.
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/402

[4] Eigen Labs. *EigenLayer Whitepaper*.
https://docs.eigenlayer.xyz/overview/whitepaper

[5] Mustafa Al-Bassam et al. *Celestia: A Scalable Data Availability Layer for Blockchains*.
https://arxiv.org/abs/1905.09274

[6] Saltzer, Reed, Clark. *End-to-End Arguments in System Design*.
https://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

[7] Verma et al. *Large-scale cluster management at Google with Borg*.
https://research.google/pubs/pub43438/

[8] MolochDAO. *DAO Governance Design and Operational Learnings*.
https://molochdao.com/

[9] NIST. *Systems Security Engineering: Considerations for a Multidisciplinary Approach*.
https://csrc.nist.gov/publications/detail/sp/800-160/vol-1/final