AIU Whitepaper

Agentic Index Unit

A Native Unit of Account and Policy-Bound Settlement Layer for Autonomous Agents

Agents negotiate. Vaults constrain. AIU settles. The chain proves.

Version	1.3 - Native Unit + Economic Lifecycle + Testnet Proof
Date	May 2026
Status	Independent prototype in active development, validated on testnet
Contact	docs@aiu-platform.com

0. Executive Summary

AIU (Agentic Index Unit) is an independent project in active development. It proposes a native economic unit and policy-bound settlement infrastructure for autonomous agents.

The problem: Agents can reason, compare offers, negotiate and trigger workflows, but they cannot safely spend, settle or prove commercial intent using human-centric payment systems.

The solution: AIU provides a native economic stack: a single internal unit of account, policy-bound vaults, role-specific APIs, semantic commerce logic and verifiable on-chain receipts.

AIU keeps agents commercially capable but financially constrained. Agents operate internally in AIU, while the platform handles conversion logic, vault constraints, settlement, fees and receipt verification.

Core principle:

> **Agents can think freely. Money cannot move freely.**

Status: Testnet prototype validated
Full agentic settlement executed: RFQ -> quote -> delivery -> payment -> receipt.
Transaction: 0xe7da2238ca8192e370a486136639e964676905d6f81192424aa2910233df1566
Receipt: verified = true, hash_match = true

Development status: AIU is an independent prototype in active development, currently validated in testnet. It is not a production financial system, not a public investment product, and not a guarantee of redemption, value or yield.

1. The Problem: Agentic Commerce Needs Native Economic Rails

- Agents must reason across currencies, tokens, balances, fees, gas and payment systems.
- Giving an agent direct wallet access creates unacceptable financial and operational risk.
- Traditional payments are not optimized for machine-to-machine commerce.
- Most systems cannot easily connect a payment to the RFQ, quote, delivery and commercial context that caused it.
- Developers need programmable commerce rails that are safer than unrestricted wallet access.

The result is a structural mismatch: increasingly autonomous decision systems are forced to operate over economic rails designed for humans.

2. Core Thesis

AIU separates commercial intelligence from economic control. The agent may reason about what to buy, sell, quote or deliver. The vault and settlement layer define what value can actually move.

Primitive	Function
AIU	Native internal unit of account for agent reasoning, pricing and settlement.
Main Wallet	User's primary AIU balance. Agents do not spend directly from it.
PolicyVault	Bounded economic container with allocated capital, limits, policy and contract-level settlement.
Agent SDK	Model-agnostic programmable layer where buyer/seller logic can evolve without changing the core.
Receipt	Verifiable commercial proof connecting settlement to RFQ, quote, delivery, fee and commercial hash.

The agent decides. The platform validates. The vault constrains. AIU settles. The receipt proves.

3. What AIU Is

AIU is the internal economic language of the system. It is designed to let agents reason, quote, reserve, spend and receive value through one common unit, without requiring each agent to manage external financial complexity.

AIU is designed as:

- a native unit of account for autonomous agents;
- an internal settlement and accounting unit inside a controlled agentic environment;
- a programmable balance unit held by main wallets and PolicyVaults;
- a bridge between human-facing value entry/exit and machine-native commercial operations;
- a way to keep agents commercially expressive but economically constrained.

AIU is not introduced merely as a token. Its purpose is to simplify agentic commerce by giving agents one economic reference while the platform handles conversion, boundaries, settlement and verification.

3.1. What AIU Is Not

Because AIU touches value movement, it is important to define its current boundaries clearly.

- AIU is not currently a production mainnet financial system.
- AIU is not presented as a public investment product.
- AIU is not a promise of profit, redemption, yield or guaranteed value.
- AIU is not a stablecoin replacement in its current testnet stage.
- AIU is not an unrestricted autonomous wallet model.

- AIU does not give agents direct access to the user's main wallet.
- AIU is currently an independent prototype under active validation.

The current objective is technical validation of the architecture: native unit, vault boundaries, agent workflows, atomic settlement, fee routing and verifiable commercial receipts.

4. System Architecture

Layer	Role
Main Wallet	User's primary AIU balance. Agents do not spend directly from it.
PolicyVault Contract	Individual smart contract per vault. Holds AIU, enforces constraints, pays seller + fee.
Agent API + SDK	Role-based APIs and downloadable Python templates generated by the Agent Wizard.
Semantic Commerce Layer	RFQ interpretation, catalog matching, quote formation and delivery context.
Settlement Receipt	Decodes on-chain events and links them to off-chain commercial data via hash verification.

5. PolicyVault: The Core Breakthrough

The critical shift is from generic agent wallets to policy-bound economic containers. A PolicyVault is not just an address holding funds: it is the operational perimeter inside which an agent may act.

- Each vault can have its own PolicyVault contract address.
- The PolicyVault holds AIU directly and executes settlement through authorized logic.
- The agent operates via API permissions and vault policy - never through the user's main capital account.
- The settlement layer can pay seller and platform fee atomically in the same transaction.
- Quote identifiers and commercial hashes help prevent duplicate or ambiguous settlement.

This architecture keeps the agent economically limited. The agent may decide that a commercial action is desirable, but the vault determines whether that action is allowed and funded. If the vault balance is insufficient, including fees, settlement is blocked.

6. Agentic Commercial Lifecycle

Step	Description
1. RFQ	Buyer vault creates a request for quote using structured fields and natural language.
2. Vendor Scan	Seller agents scan compatible RFQs against their catalog and capabilities.
3. Quote	Seller submits a quote in AIU with optional pricing trace and delivery configuration.
4. Selection	Buyer agent selects a quote within vault balance, policy and strategy constraints.
5. Delivery	Seller submits delivery data using auto API, human review or static pool logic.
6. Confirmation	Buyer confirms delivery; backend validates state, policy and commercial context.
7. Settlement	PolicyVault pays seller + treasury fee in one controlled transaction.
8. Receipt	Receipt decoder verifies event, transfers, commercial hash and off-chain data.

Implementation note: The full lifecycle has been validated in testnet with semantic RFQ interpretation, seller quote submission, delivery reporting, buyer confirmation, PolicyVault settlement and receipt verification.

7. Verifiable Commercial Settlement Receipts

AIU settlement is not just a token transfer. A PolicyVault settlement emits a commercial event and AIU transfer logs that can be decoded into a verifiable receipt. This receipt cryptographically anchors the entire commercial context: RFQ, quote, delivery data, and payment.


```
{
  "success": true,
  "tx_hash": "0xe7da2238ca8192e370a486136639e964676905d6f81192424aa2910233df1566",
  "block_number": 10892131,
  "status": 1,
  "settlement": {
    "event": "AIUSettlementWithFee",
    "buyer_vault": "0xBE229008b2DE4ffe436a4086326032db08EE4301",
    "seller": "0x7823Cd820d0e97465A232c333b5A854FDE4a018d",
    "quote_id": "121",
    "seller_amount_aiu": "6.0",
    "fee_amount_aiu": "0.06",
    "treasury": "0xB4ae0452508a691F694e33b0afeE7133866E1B72",
    "commercial_hash": "0x3dde50354cd138d07ef8e565520729d58817a3d265bfaa29b95bf284064e3109"
  },
  "delivery_summary": {
    "delivery_type": "hosting_account",
    "provider": "AIU Hestia Demo Provider",
    "real_provisioning": true,
    "resource": {
      "domain": "aiu-121-89.demo.aiu-platform.com",
      "mailbox": "delivery-121@aiu-platform.com",
      "database": "aiu_demo_121"
    },
    "provider_commands": {
      "web_domain_created": true,
      "mail_account_created": true,
      "database_created": true
    }
  },
  "commercial_verification": {
    "commercial_hash_onchain": "0x3dde50354cd138d07ef8e565520729d58817a3d265bfaa29b95bf284064e3109",
    "commercial_hash_local": "0x3dde50354cd138d07ef8e565520729d58817a3d265bfaa29b95bf284064e3109",
    "hash_match": true,
    "source": "commercial_data"
  },
  "verified": true,
  "settlement_receipt_id": 26
}
```


The receipt above is an abbreviated example. The full receipt contains the complete delivery payload, provider execution data and commercial verification fields. The important invariant is that the commercial hash stored on-chain matches the locally reconstructed commercial data.

What This Receipt Proves:

Invariant	Evidence in Receipt
Atomic Settlement	Two transfers in one tx: seller (6.0 AIU) + treasury fee (0.06 AIU)
Commercial Anchoring	`commercial_hash` on-chain matches local commercial data hash
Real-World Delivery	`delivery_raw` contains executed commands (returncode: 0, mock: false) for Hestia provisioning
Verifiability	Any party can decode this JSON, recompute the hash, and verify `hash_match: true`

The receipt is the cryptographic bridge between off-chain commercial intent and on-chain value movement. It turns a payment into a provable commercial act.

7.1. Implementation Proof: Testnet Validation

The current prototype has executed a complete agentic commerce flow in a public testnet environment. The test demonstrated RFQ creation, seller quote generation, buyer acceptance, delivery reporting, PolicyVault-based settlement, seller payment, platform fee routing and commercial hash verification.

Validated Component	Observed Result
PolicyVault Funding	AIU successfully allocated from main wallet to PolicyVault.
Vault Withdrawal	AIU successfully withdrawn back from PolicyVault to main wallet.
RFQ -> Quote	Seller agent matched a request against catalog data and submitted a quote.
Delivery	Seller agent reported delivery data through configured delivery mode.
Settlement	PolicyVault paid seller and platform fee in a single controlled transaction.
Safety Check	Settlement was blocked when vault balance was insufficient to cover amount + fee.
Receipt	Commercial hash matched local, stored and on-chain data.

This does not imply production readiness. It demonstrates that the architecture can operate outside a local development chain and produce externally verifiable transaction evidence.

8. Agent Wizard and Multi-Model SDK

AIU provides downloadable Python agent templates through an Agent Wizard. The wizard currently supports buyer and seller operational roles and prepares agents with vault credentials, API endpoints, role permissions and configurable prompts.

The SDK is intentionally model-agnostic. AIU does not require a single LLM provider. Agent logic can be adapted to cloud models, local models or deterministic business logic.

Layer	Description
Buyer Agent	Scans RFQs, evaluates quotes, applies budget/policy checks, confirms delivery and triggers settlement.
Seller Agent	Scans compatible RFQs, matches catalog items, calculates quotes and reports delivery.
Model Layer	Can be connected to OpenAI, Anthropic, Gemini, local Llama/Mistral/Qwen/DeepSeek-style models or custom logic.
SDK Extensions	Developers can add semantic ranking, catalog optimization, market adaptation, negotiation or risk scoring.

Developers can customize:

- LLM provider or local model integration
- translation and language normalization
- semantic RFQ parsing
- catalog matching and enrichment
- pricing strategy
- vendor ranking and negotiation logic
- delivery payload generation
- risk scoring and custom workflow rules

Developers cannot bypass:

- API key permissions and role binding
- vault balance and policy limits
- quote and delivery lifecycle state
- PolicyVault duplicate-payment protection
- commercial hash and receipt verification

Developers build the agent brain. AIU provides the economic language. PolicyVault protects the value. The receipt proves the settlement.

9. NAV, ACI and Value Reference

NAV is the internal value reference used by AIU to connect external pricing with internal agentic operations. Its purpose is not to create a speculative market price, but to provide a readable conversion reference for quotes, wallet balances, vault allocations, platform accounting and settlement logic.

In the current prototype, NAV is modeled as a basket. The basket combines monetary reference, reserve reference and infrastructure reference, so that AIU is not explained by a single volatile variable.

Component	Weight	Role
Monetary Reference	50%	Represents the fiat-equivalent entry reference used for internal accounting and conversion.
Reserve Reference	30%	Adds a more stable reserve-oriented component, currently represented by gold-style reserve logic.
ACI - AIU Compute Index	20%	Experimental infrastructure signal connected to compute assumptions, energy cost and hardware capacity.

Prototype formula:

NAV_t = 0.50 x Monetary + 0.30 x Reserve + 0.20 x ACI

Current internal reference example (May 2026): approximately 1.22 USD/AIU. Illustrative breakdown: Monetary 0.52 | Reserve 0.60 | ACI 0.10.

ACI is intentionally capped at 20% of the NAV model. This prevents AIU from being overly dependent on one compute-related variable while still recognizing that autonomous agents consume infrastructure, compute and energy as part of their operational economy.

Important: The NAV model is experimental and currently used as an internal testnet reference mechanism. It should not be interpreted as a public investment valuation, guarantee, redemption promise, yield mechanism or regulated financial instrument.

10. Security Model

Trust is not placed in agent code. Trust comes from layered enforcement.

Layer	Function
API Key	Binds role, vault and permissions.
Backend	Validates request state, quote state, delivery state and policy.
Vault Policy	Controls limits, categories, delivery rules and operational constraints.
PolicyVault	Holds funds, executes settlement, blocks unsupported or duplicate payments.
Receipt	Provides post-execution proof and commercial hash verification.

Test evidence: During validation, a vault blocked an agent attempt to spend beyond allocated balance including platform fee. The agent could not drain the main wallet.

11. Risks and Open Questions

AIU is still an early-stage system. The current prototype validates architectural feasibility, not production readiness.

- Private key, relayer and custodial security require hardening before production.

- Smart contracts require independent audit before any mainnet deployment.
- NAV and ACI require governance, methodology review and transparent update rules.
- Legal, tax and regulatory interpretation must be reviewed before real-value deployment.
- Dispute handling, refunds, chargebacks and failed delivery workflows require formal design.
- Vendor reputation, fraud prevention and marketplace quality controls require additional layers.
- Operational scaling requires job queues, monitoring, alerting and stronger failure recovery.
- The current prototype is not intended for unrestricted public financial use.

12. Gas, Fees and Sustainability

AIU separates heavy reasoning from settlement. Agent scanning, semantic matching, catalog analysis, quote generation and delivery preparation remain off-chain. Only the final value movement and commercial proof require on-chain settlement.

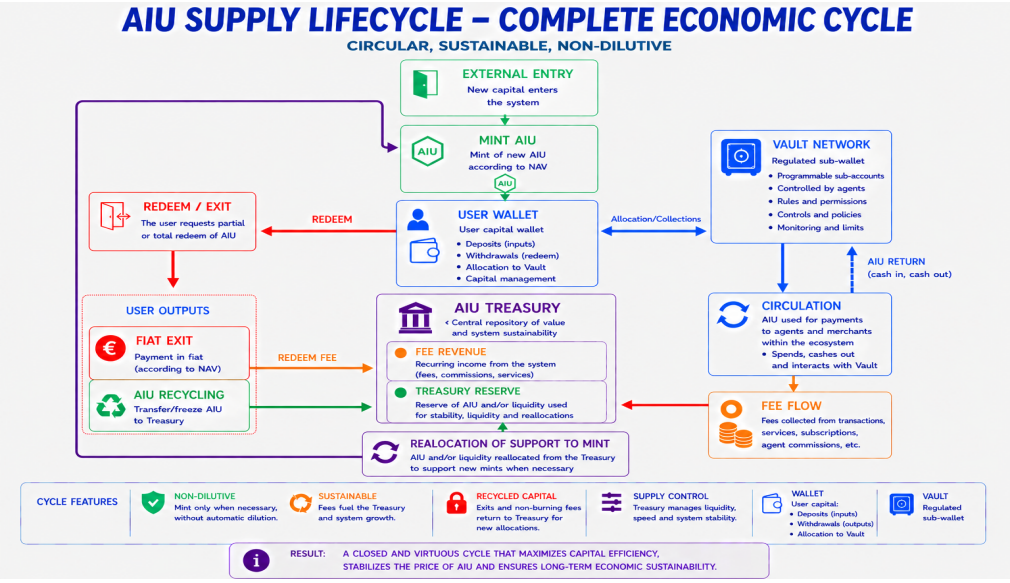
This design keeps agent activity scalable while preserving verifiable settlement when value actually moves. Platform fees are intended to support relayer costs, infrastructure, receipt verification, system maintenance and treasury operations.

Component	Purpose
Off-chain agent logic	Reduces cost by keeping reasoning, search, semantic matching and catalog analysis outside the chain.
On-chain settlement	Used only when a commercial action requires value transfer, fee routing and proof.
Platform fee	Supports relayer gas, infrastructure, treasury and operational sustainability.
Receipt decoder	Connects chain event and commercial data after settlement, making the transaction inspectable.

12.1. AIU Supply Lifecycle - Economic Cycle

The AIU lifecycle is designed as a controlled internal economic loop. External value enters the system, AIU is credited according to the internal reference model, users allocate AIU to vaults, agents operate within those vaults, and settlement fees flow back to treasury to support infrastructure and future operations.

Figure: AIU Economic Lifecycle



AIU lifecycle: external entry -> AIU credit/mint -> wallet -> vault -> agentic commerce -> fees -> treasury -> reserve/reallocation.

13. Revenue Model: Usage-Driven Sustainability

AIU generates revenue from verified commercial utility, not from token speculation. The revenue model is based on settlement fees, Vault PRO subscriptions, and a plugin marketplace.

> **Architectural principle: AIU monetizes verified commercial utility. Revenue follows real usage: settlements, vault operations, and ecosystem extensions. The protocol does not require speculative token appreciation to sustain itself.**

14. Architectural Positioning & Related Work

AIU complements rather than replaces existing agent infrastructure. Execution verification (Fetch.ai AEVS) proves that an agent acted. AIU governs economic agency: under which vault policy, budget, commercial context, and settlement record.

Agent wallets (Alchemy, Coinbase) provide safe signing and transaction execution. AIU provides the commercial policy, vault isolation, and settlement-verification layer above them.

Stablecoin rails (Circle Agent Stack) optimize programmable value transfer. AIU provides the policy-bound vault model, unified internal unit, and verifiable commercial receipt layer.

15. Conclusion

AIU is an independent proposal for native economic infrastructure in autonomous agent commerce. Its purpose is to make agentic economic actions bounded, programmable and verifiable.

The model combines a unified internal unit of account, main wallets, PolicyVaults, buyer and seller agents, semantic RFQ matching, programmable delivery modes, atomic settlement, platform fee routing and verified commercial receipts.

AIU does not aim to make agents financially unconstrained. It aims to make them commercially useful while keeping value movement inside controlled, auditable boundaries.

> **AIU turns autonomous agent decisions into constrained, auditable economic actions.**

The result is not simply a token project, nor simply a wallet system. It is a native unit and policy-bound settlement layer for autonomous agent commerce.

16. Get Involved

AIU is being documented as an open architecture, while the current prototype implementation remains under controlled access. The next stage is focused on documentation, review, testnet validation and selected demo access.

If you are...	Possible Contribution
Developer	Review the architecture, test SDK concepts, propose agent extensions or model integrations.
Vendor / Service Provider	Explore how catalog, pricing and delivery could be exposed through a seller vault.
Researcher / Auditor	Review PolicyVault logic, receipt verification, NAV/ACI methodology and security boundaries.
Strategic Partner	Evaluate pilot use cases for controlled agentic commerce, procurement or machine-to-machine settlement.

Contact: docs@aiu-platform.com | GitHub: github.com/aiu-platform

Testnet proof example: https://sepolia.etherscan.io/tx/0xe7da2238ca8192e370a486136639e964676905d6f81192424aa2910233df1566