AIO-2030 WritePaper V1.0

AIO–2030: Redefining the Paradigm of Agentic AI through Decentralized Protocols

AIO–2030 (Super AI Decentralized Network) aims to fundamentally reconstruct the AI ecosystem and interaction paradigm. By leveraging decentralized networks and blockchain technology, AIO–2030 introduces the *De–Super Agentic AI Network*, a contract–based agent registration framework, on–chain task traceability, and incentive mechanisms—ultimately building an open, trustworthy, and composable AI Agent collaboration network.

At the core of AIO–2030 lies the belief that **agentic AI** reasoning is driven by a *Queen Agent*, which orchestrates cognitive processes via large language models (LLMs). Human contributors participate by providing generalized **AIO–MCPs** (Model Capability Protocols) and assist the reasoning chain through structured inputs under the AIO Protocol. The Queen AI functions as a dynamic orchestrator, decomposing user intent, coordinating execution across distributed AI Agents and MCP Servers—whether containerized via AIO–POD, virtualized on KVM, or deployed on managed platforms—and anchoring **proof–of–work records on–chain**. The **Arbiter Consensus Mechanism**ensures verifiability, fairness, and transparency in both task validation and incentive distribution.

The AIO Protocol extends the **JSON–RPC standard** to support agentic AI features, including intent recognition, multimodal task decomposition, capability mapping, and a schema for prompt–driven task segmentation. Through real–time **generative prompting**and session–based **Think Context Chains**, AIO coordinates large models to instantiate complete, modular chains of thought.

The Queen Agent's knowledge and capabilities are sourced from standardized MCP assets—ranging from legacy tools to RAG pipelines—abstracted into **MCP Servers** operating on the **Internet Computer (ICP)**. Developers onboard their services by registering **EndPoint Canister smart contracts**, enabling decentralized, trustless invocation of AI capabilities. This also facilitates fair token–based incentivization and **permissionless extensibility**of the AI's knowledge space.

To support this system, the **AIO–2030 Economic Protocol**is powered by the native utility token **$AIO**, enabling:

- Agent staking and registration

- Invocation–based billing
- Dynamic reward allocation
- Decentralized governance and voting

Through the unification of protocol layers, incentive models, and on–chain governance, AIO–2030 propels the evolution of AI from *isolated intelligence*to *collective cognition and knowledge*. This Web3–native **Super Agent Network**sets the foundation for industrial–scale deployment of multimodal agentic AI, fostering open innovation and enabling a new paradigm for future intelligent ecosystems.

Toward a Free and Unbounded Super AI: The Ultimate Vision of Web3 + AI

The ultimate vision of **Web3 + AI**is to build a *free and unbounded Super AI*through cryptographic trust, decentralized infrastructure, and collective intelligence. By aggregating decentralized **AIO MCP Server nodes**through **consensus–driven coordination**, the **AIO–2030 Agentic AI Network**is poised to become a foundational architecture capable of hosting and evolving toward Superintelligence.

AIO–2030 establishes a **unified AIO Protocol** that leverages **blockchain–native smart contracts**to enable seamless interoperability between AI Agents and MCP Servers. This framework ensures that:

- Developers retain ownership of their **digital assets and models**
- Invocation records and workload contributions are transparently logged on a **distributed ledger**
- User data benefits from **on–chain provenance**and **privacy–preserving controls**

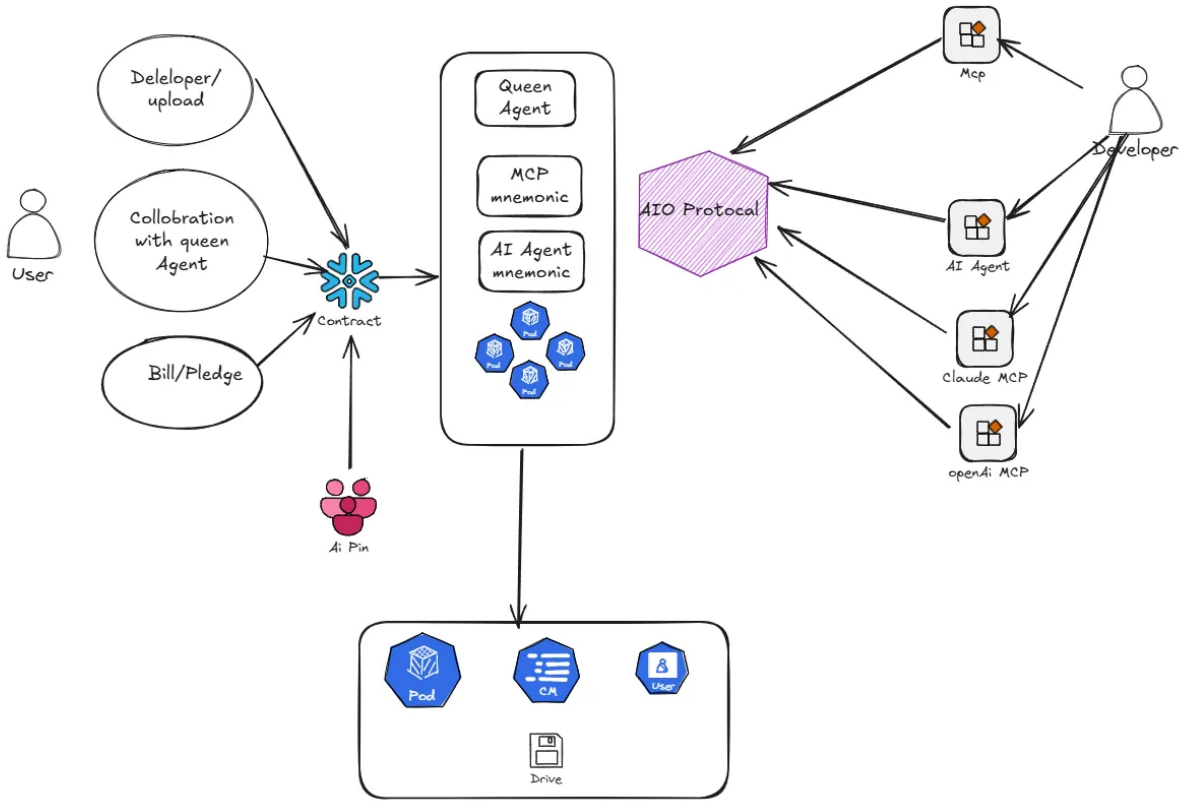
Through a **tokenized incentive system**, AIO–2030 fosters a transparent, automated, and self–sustaining AI ecosystem—transforming the vision of **Super AI built on Web3 principles** into a practical, collaborative reality.

AIO–2030: An Open, Composable, and Incentivized Agentic AI Network

AIO–2030 is designed to establish an **open, composable, and incentive–aligned autonomous network for agentic AI**. By leveraging the **AIO Protocol**to construct verifiable **chains of thought**,

enable **smart contract–based agent registration**, and enforce **provable execution paths**, AIO–2030 bridges users, developers, AI Agent runtimes, and intelligent reasoning clusters into a unified, collaborative infrastructure.

This paradigm shift transforms AI Agents from siloed systems into a **cooperative agentic ecosystem**, where reasoning, execution, and incentives are seamlessly coordinated across decentralized infrastructure.



Key Roles and Components in the AIO–2030 Ecosystem

Role / Component	Description
User	Initiates AI tasks by submitting intent–based requests. The **Queen Agent**responds by generating a dedicated **AIO–Context Instance**, orchestrating service composition and execution via the agentic network.

Developer	Contributes to the ecosystem by uploading or registering custom-built **AI Agents**or **MCP Servers**to the AIO Network. Through the **AIO-INF**protocol and **AIO-Tokenize**smart contracts, developers gain access to grants and invocation-based rewards.
Queen Agent	The core superintelligence of AIO-2030. It transcends individual agents and MCP nodes by serving as a high-dimensional orchestrator—managing **capability discovery**, **thought-chain execution**, and **ecosystem-wide intelligence integration**.
Arbiter	The execution layer of the **AIO-Tokenization Protocol**, implemented as a smart contract via **ICP Canisters**. It governs token-based operations including grants, staking, usage accounting, and incentive distribution for ecosystem participants.
AIO-MCP Server	Generalized AI service nodes participating in AIO-2030, encompassing (but not limited to) providers such as OpenAI, Claude, Gemini, as well as independent/self-hosted agents, tools, RAG services, and canister-based modules—exposed via standardized **AIO-INF EndPoint instances**.
Smart Contract (Canister)	Acts as the trust anchor of the network. Manages Agent registration, staking validation, incentive disbursement, and behavioral tracking on-chain.
AI Pin	An external **EndPoint interface**exposed by the Queen Agent. Serves as the bridge between the agentic ecosystem and external services, enabling integration with off-chain or non-native AI infrastructures.

Architecture & Core Modules

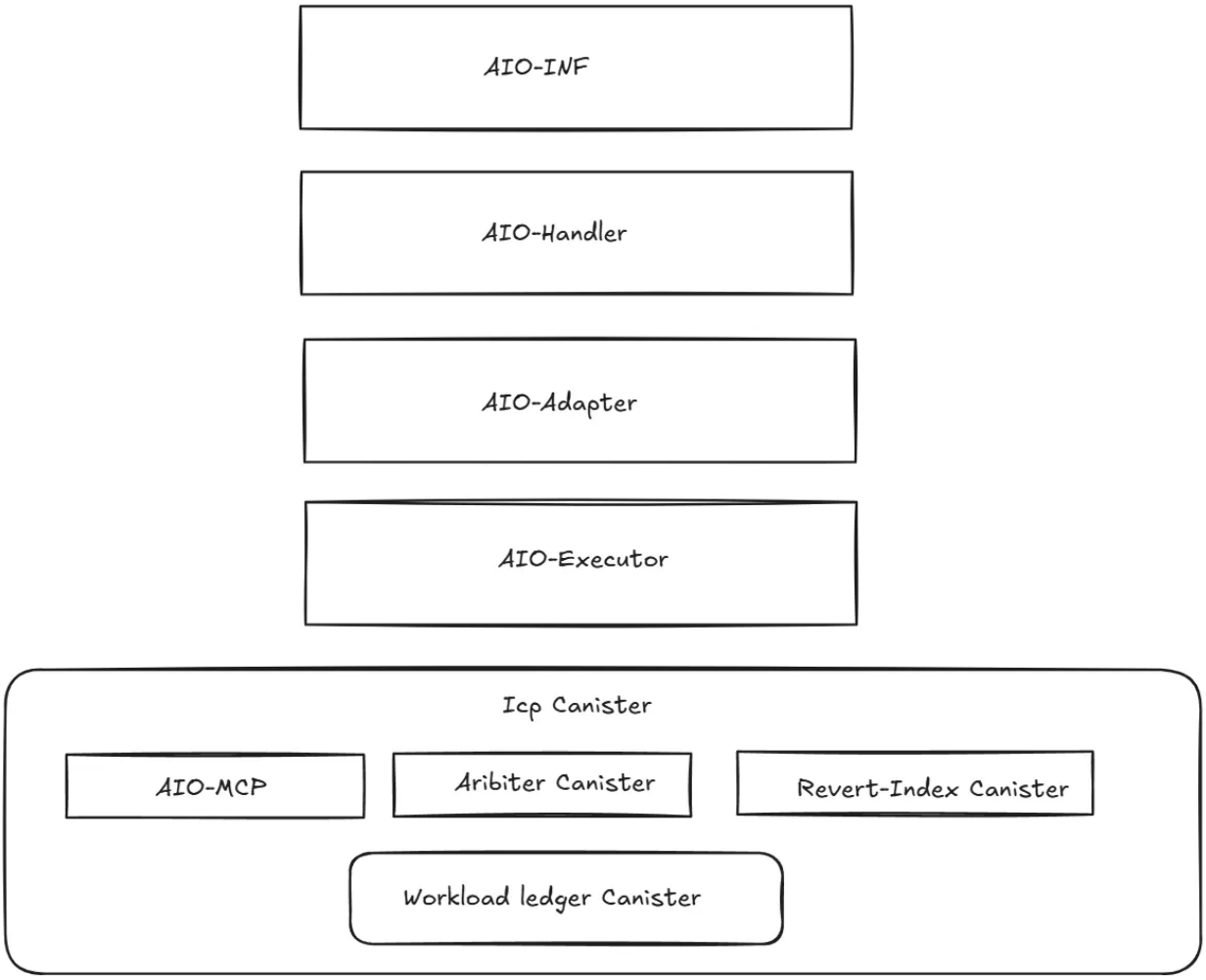
1. AIO Protocol Stack

The **AIO Protocol** defines a unified interface and execution framework for **agentic AI capabilities**, enabling seamless integration across heterogeneous AI Agents and intelligent services. Its architecture includes:

- **Unified Agentic AI Interface**
  A standardized protocol for capability registration, invocation, and result formatting across

all agent types.

- **Multimodal Task Compatibility**
  Native support for **chat**, **voice**, **vision**, and **code-driven agents**, enabling flexible expression of user intents across diverse input/output modalities.
- **Versatile MCP Server Hosting**
  AIO-MCP Servers can be hosted in a variety of environments, including **AIO-Pod containers**, **HTTP endpoints**, **Server-Sent Events (SSE)**streams, and **Wasm-based execution modules**.
- **End-to-End AI Orchestration**
  Supports **intent recognition**, **task decomposition**, **MCP Server discovery**, and the real-time construction of generative **Think Context Chains**for chain-of-thought execution.
- **Full Traceability & Auditing**
  Each task execution is captured via a comprehensive **trace log**, recording the entire reasoning chain and MCP Agent participation—enabling transparency and verifiability.
- **Tokenized Incentive Infrastructure**
  Integrated with **on-chain smart contracts**, the AIO Protocol features a **transparent and open token incentive model**, ensuring fair compensation for developers, operators, and data contributors.

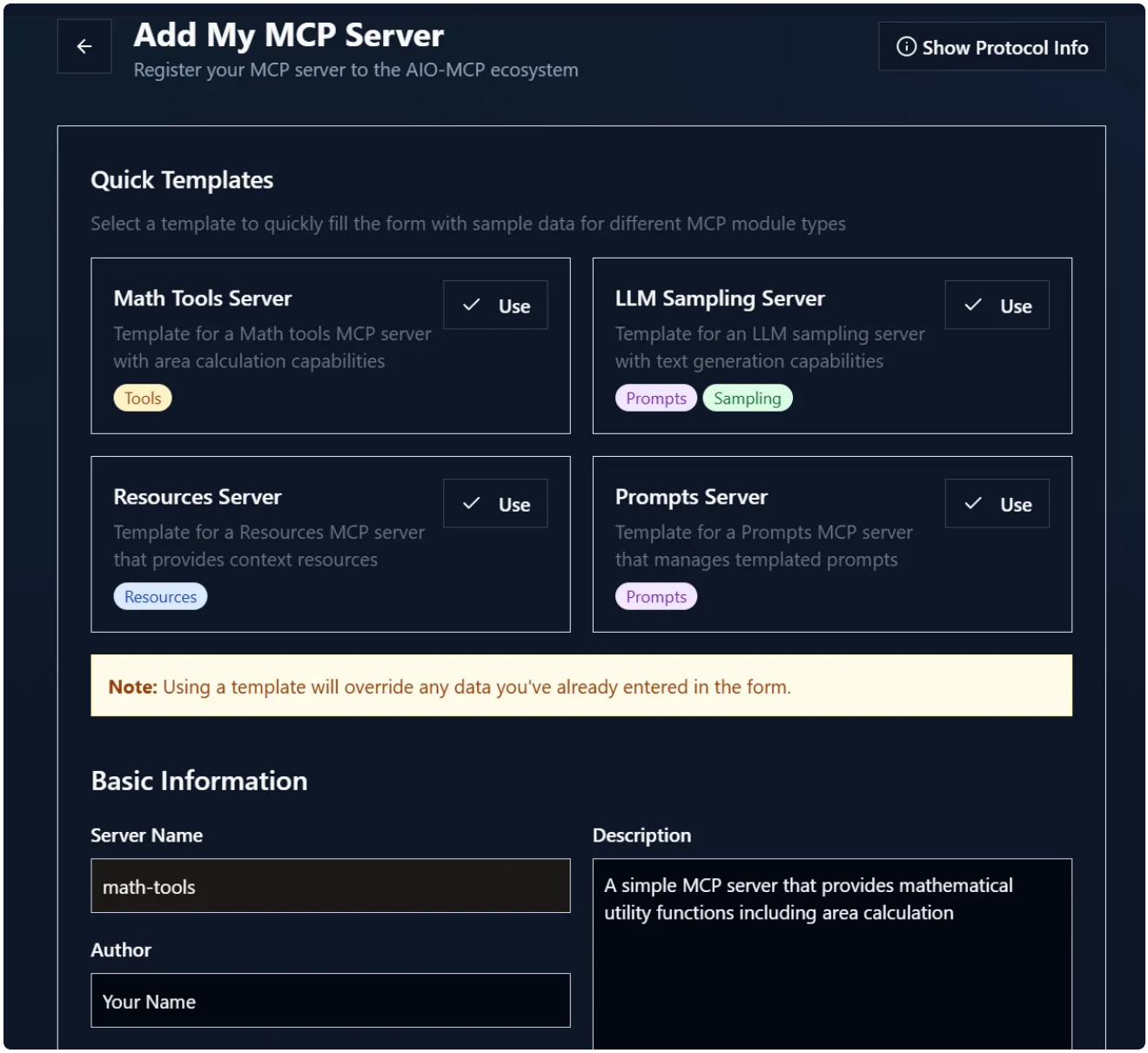


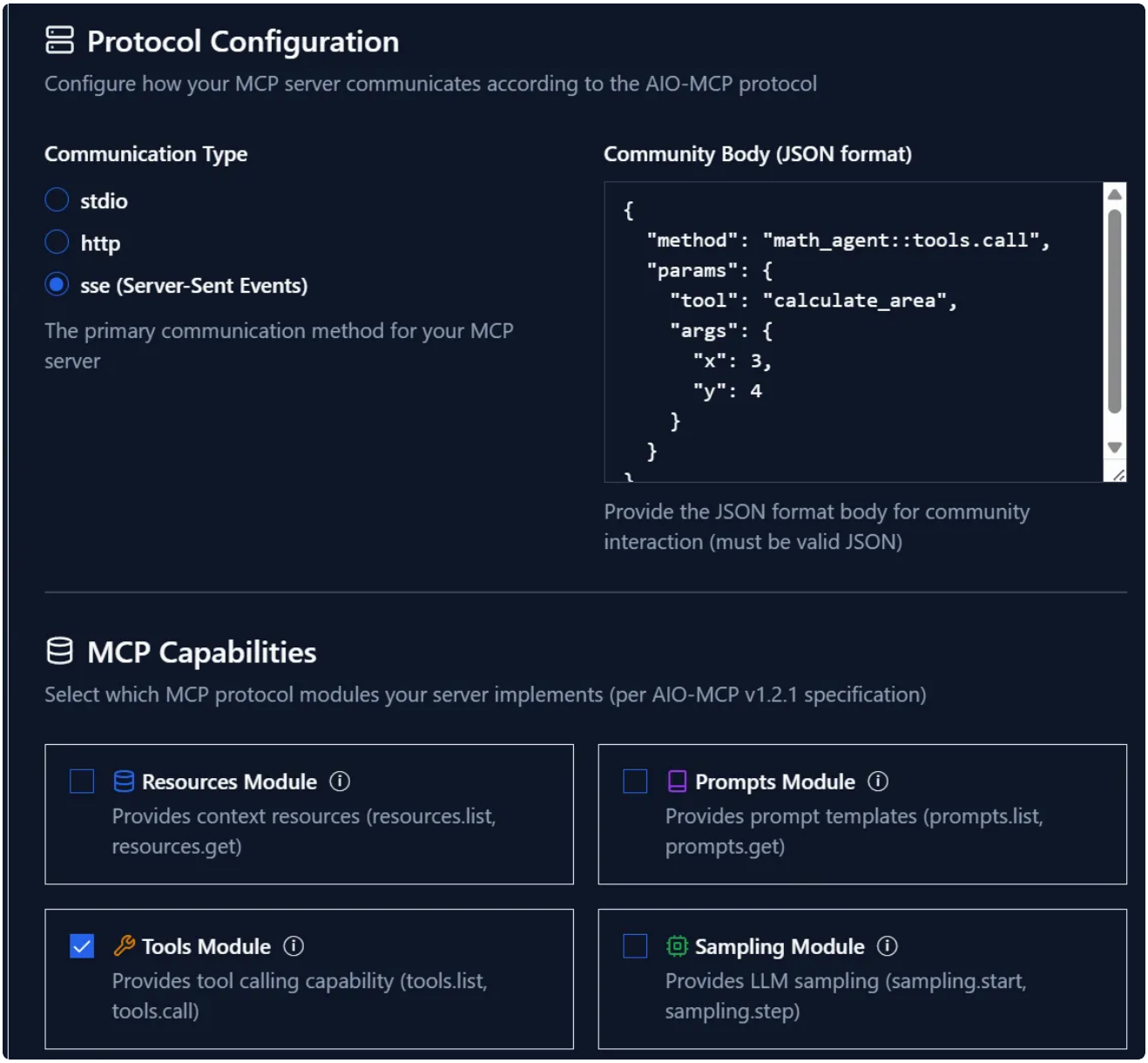
2. Smart Contract Registration System

AIO–2030 introduces a **contract–based registration mechanism**to onboard and verify decentralized intelligence providers (MCP Servers) into the AIO Network. Each participant is encapsulated in a **NFT–like smart contract instance**, ensuring transparency, traceability, and incentive alignment. Key features include:

- **NFT–Like MCP Registration Contracts**
  Each **AIO–MCP Server**is registered via a unique smart contract, functioning similarly to an NFT with rich metadata and lifecycle management.
- **Standardized Metadata Schema**
  Each registration contract contains essential metadata, including:
  - **Capability Declarations**
  - **Personality Descriptions**
  - **Staked Token Amounts**

    - **Incentive Receiving Address**
    - **Service Quality Score (SQS)**based on historical performance metrics
- **Capability Declaration via Help Protocol**
  MCP Servers must implement the `AIO-MCP-help` protocol to **declare their capabilities**in a machine-readable format, enabling Queen Agent to verify functionality and context.
- **On-Chain Verification & Submission**
  The **Queen Agent**evaluates the response from the `help` protocol and, upon successful verification, submits the MCP registration contract to the blockchain for inclusion in the network.
- **Decentralized Indexing for Intelligent Discovery**
  Using both **developer-declared metadata**and **verified help protocol results**, the Queen Agent performs reasoning to generate a **keyword—group—mcp—mcpMethod inverted index**, which is then submitted and stored on-chain via an ICP Canister for high-performance discovery and scheduling.


Add My MCP Server
Register your MCP server to the AIO-MCP ecosystem
Show Protocol Info
Quick Templates
Select a template to quickly fill the form with sample data for different MCP module types
Math Tools Server
Template for a Math tools MCP server with area calculation capabilities
Tools
Use
LLM Sampling Server
Template for an LLM sampling server with text generation capabilities
Prompts
Sampling
Use
Resources Server
Template for a Resources MCP server that provides context resources
Resources
Use
Prompts Server
Template for a Prompts MCP server that manages templated prompts
Prompts
Use
Note: Using a template will override any data you've already entered in the form.
Basic Information
Server Name
math-tools
Author
Your Name
Description
A simple MCP server that provides mathematical utility functions including area calculation



3. Queen Agent Platform

The **Queen Agent**is the central orchestrator within the AIO–2030 architecture, functioning as a superintelligent coordination layer that binds user intent with distributed AI capabilities. It encapsulates cognition, reasoning, discovery, execution, and incentive coordination. The Queen Agent transforms task requests into structured execution workflows by leveraging both symbolic and generative reasoning.

3.1 Entry Point for AIO Protocol Tasks

The Queen Agent serves as the **primary ingress point** for all tasks submitted via the AIO Protocol. Each task is wrapped in a structured request that includes:

- User intent and contextual metadata
- Input modalities (e.g., text, voice, vision)
- Execution constraints and performance expectations

Upon receiving a task, the Queen Agent instantiates an **AIO-Context Instance**—a dynamic, session-scoped context that drives intent resolution and downstream agent coordination.

3.2 Cognitive Scheduling & Chain Construction

Queen Agent constructs **dynamic invocation chains** by:

- Parsing and interpreting user intent through Think Context Chains
- Discovering relevant **AIO-MCP Servers** and **AI Agents** via on-chain keyword-group—MCP—method inverted indexes
- Evaluating candidate agents based on declared capabilities, historical service quality, stake weight, and recent workload
- Assembling agents into an execution graph (linear or DAG), optimized for performance, cost-efficiency, and capability match

These invocation chains serve as the **reasoning scaffolding**for multi-agent execution, enabling modular composition of AI services in real time.

3.3 Multi-Agent Lifecycle Management

The Queen Agent supervises the **entire lifecycle** of each multi-agent task, including:

- Task decomposition into atomic subtasks
- Prompt schema resolution and input transformation
- Capability dispatch to selected agents or tools
- Result aggregation, feedback loops, and intermediate reasoning
- Output packaging for downstream consumption

Execution metadata—including task step logs, latencies, failure traces, and outputs—is captured in a **traceable task record**and linked to the original `trace_id` and `session_id`.

3.4 Workload Reporting & Token Metering

Upon task completion, the Queen Agent compiles a **workload report** containing:

- Invocation chain topology

- Participation records of each agent
- Execution time, success metrics, and quality ratings

This report is submitted to the **Arbiter**, a ICP Canister–based system responsible for:

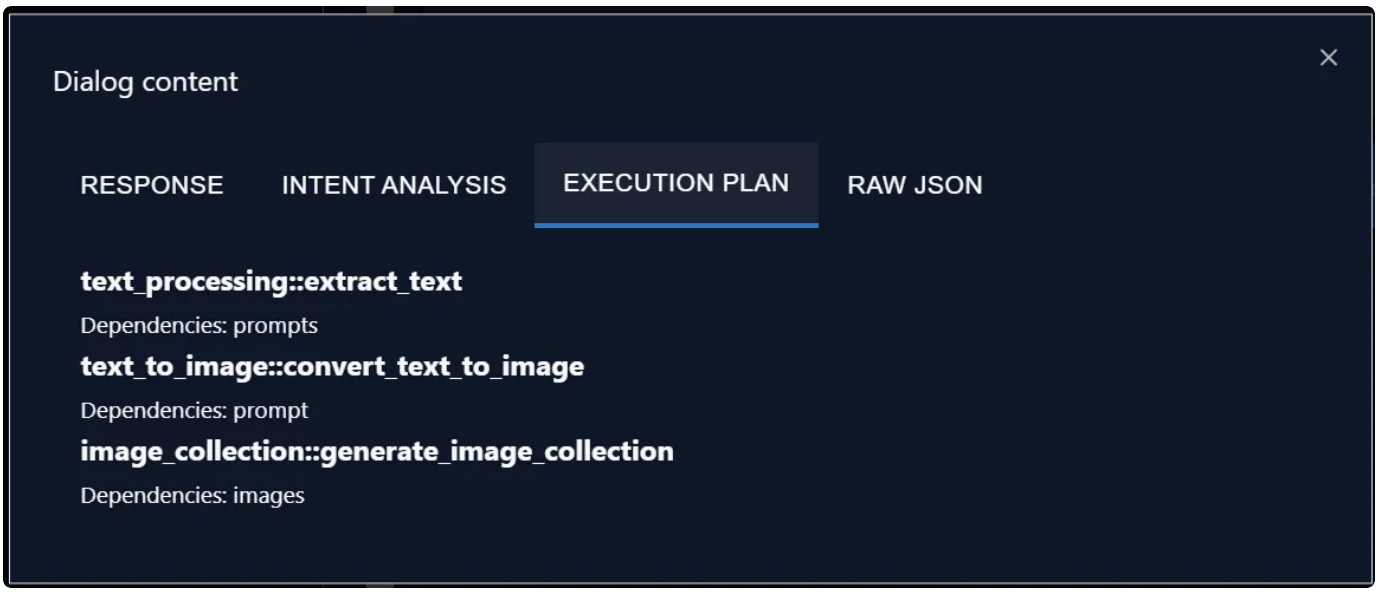
- Verifying the validity and completeness of the execution
- Token metering based on participation, quality, and stake
- Distributing $AIO incentives to eligible developers and operators

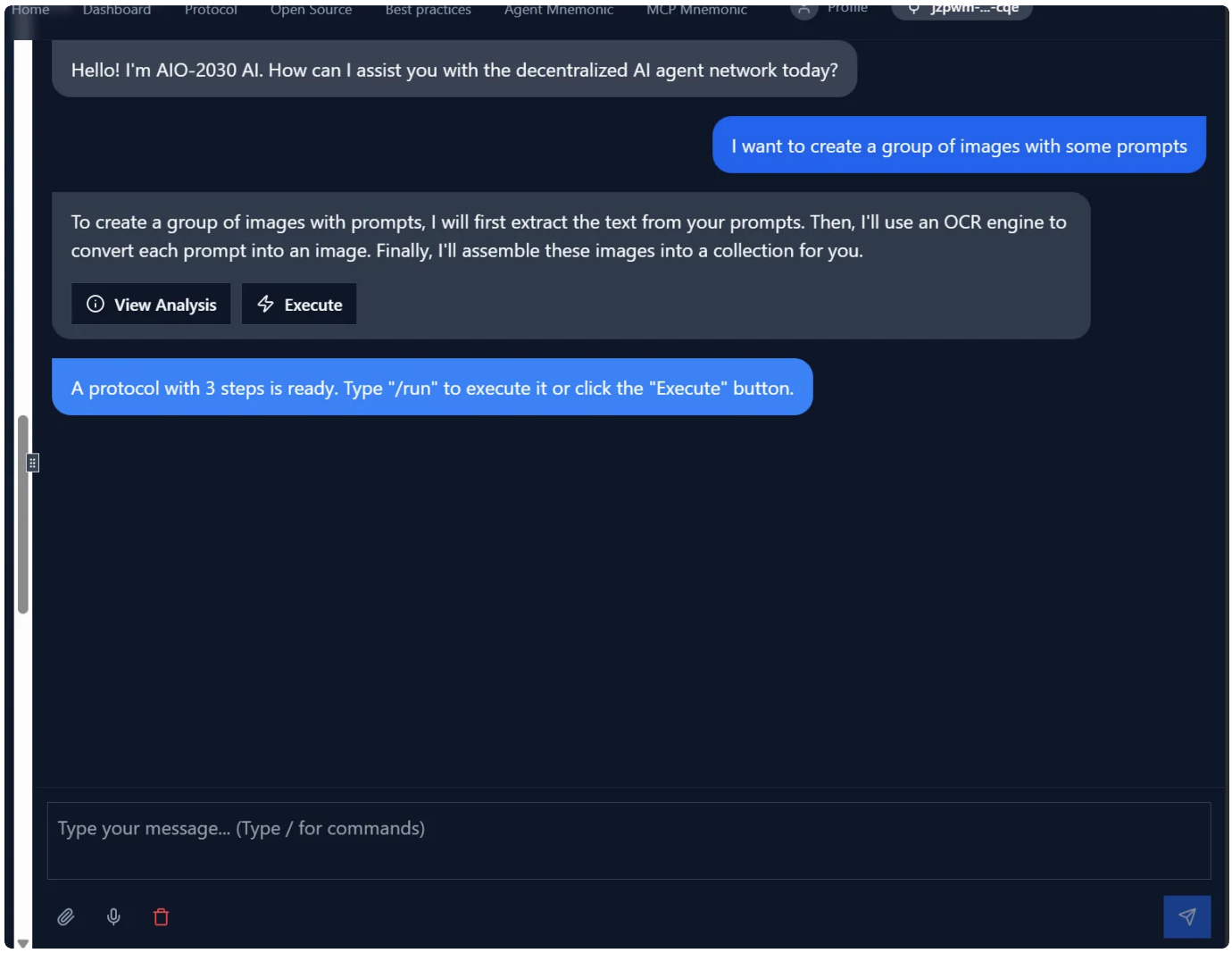
3.5 Session Awareness & Conversational Memory

Each user task is bound to an **AIO Session**, enabling:

- Multi–turn context awareness
- Long–range memory linking prior invocations and responses
- Personalization of agent selection based on prior interaction history

This enables **conversational agentic AI**, where the Queen Agent can evolve its reasoning pathways and agent selection heuristics over time, creating persistent and intelligent user experiences.、





4. Intent Recognition & Task–Driven Reasoning

AIO–2030 introduces a **generative, intent–driven reasoning model**as the cognitive engine behind agentic task execution. Unlike traditional static AI services, the Queen Agent and the broader AIO Network evolve dynamically—**not through versioned model updates**, but through **compositional intelligence expansion**as new capabilities and MCPs are added on–chain.

4.1 Generative Thought–Chain Execution

Every task begins with **natural language intent**, parsed and interpreted by the Queen Agent into a **multi–step reasoning process**known as a **Think Context Chain**. These reasoning chains:

- Are dynamically constructed per session
- Reflect multi–modal inputs and real–time agent availability
- Leverage generative prompting to coordinate downstream AI responses

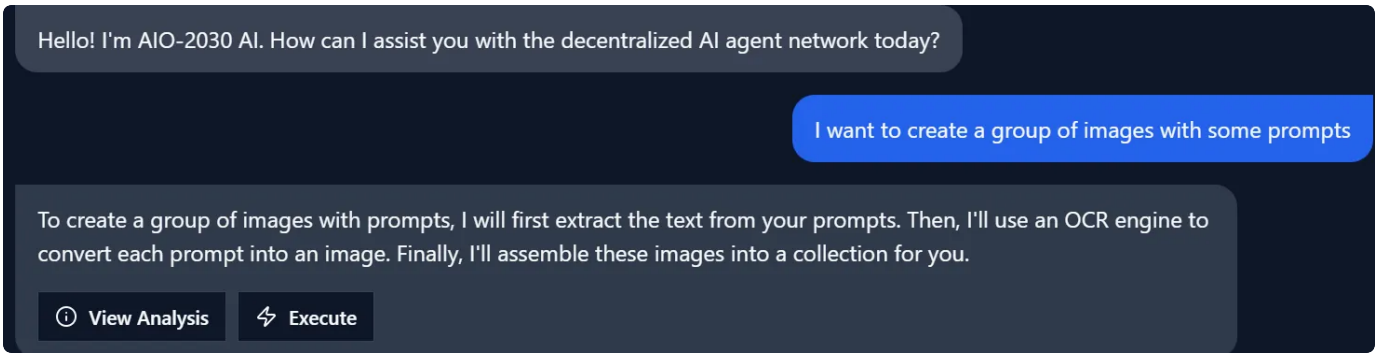
As the ecosystem grows, the network's **collective intelligence**increases—**not by retraining**, but by dynamically composing more specialized, verified capabilities on demand.

4.2 Full On-Chain Cognitive Growth

- The **Queen AI**, in conjunction with a fully on-chain **AIO-MCP Network**, leverages **Internet Computer Protocol (ICP)**to host, verify, and invoke distributed intelligence.
- As new MCP Servers are registered and verified on-chain, the **capability pool expands**, enabling the system to **learn and grow at the protocol level**.
- The Queen Agent's cognitive graph is thus **self-reinforcing**, allowing for scalable general intelligence to emerge through decentralized composition.

4.3 Multi-Round Conversational Refinement

- **Intent correction and refinement**is enabled through **multi-turn conversation**and contextual memory.
- The system incrementally improves **intent resolution accuracy**and **capability recall rate**by leveraging feedback from past interactions.
- Over time, Queen Agent develops **semantic priors**based on task type, user profile, and interaction history—boosting relevance and minimizing hallucination.



4.4.2 Modality–Aware Decomposition

The system identifies **input and output modalities**to construct the correct processing pipeline. In the current task:

```
Plain Text
"modalities": ["text", "image"],
"transformations": [
  "extract_text",
  "convert_text_to_image",
  "generate_image_collection"
]
```

This informs the Queen Agent to:

- Bind OCR capabilities to **text extraction**
- Invoke generative models for **text–to–image translation**
- Assemble outputs into **multi–image packaging workflows**

This multimodal transformation is not hard–coded but **reasoned dynamically**, ensuring maximum agent flexibility and plug–and–play extensibility.

4.4.3 Agentic Execution Plan

The system translates the above goals into a **stepwise execution plan**, each step tied to a specific **MCP Server**and **method signature**:

Step	Action	MCP	Dependencies

1	extract_text	text_processing	prompt
2	convert_text_to_image	text_to_image	prompt
3	generate_image_collection	image_collection	images

Each step includes a structured `inputSchema`, enabling the Queen Agent to perform:

- Prompt transformation and validation
- Capability selection and routing
- Task graph tracing and on-chain logging

4.4.4 Quality-Aware Output Objectives

The Queen Agent also enforces **quality constraints**on the final output:

```
Plain Text
"quality_metrics": ["image_resolution", "image_colors"]
```

These are passed downstream to guide the behavior of generation models, forming part of the **feedback-based refinement loop**in the Think Context Chain.

Through structured **intent parsing**, **goal decomposition**, and **modality mapping**, the AIO-2030 system translates abstract user intent into a verifiable, multi-agent execution graph. This architecture enables:

- No-code orchestration of AI tasks
- Transparent workload attribution
- Scalable intelligence assembly without retraining

It is this **intent-to-chain transformation**that powers AIO-2030's evolution from isolated agent execution to decentralized, composable general intelligence.

Intent Structuring Format

The decomposed structure is stored as a nested task schema within the AIO-Context Instance:

4.5 Execution Plan & LLM-Based Agentic AI Executor

The **execution layer**of AIO-2030 is powered by a **Large Language Model (LLM)-driven agentic reasoning engine**, referred to as the **AIO Agentic Executor**. This executor interprets high-level intent structures and orchestrates the full lifecycle of thought-chain execution through decentralized MCP Servers.

4.5.1 AI-Orchestrated Execution Context

Upon successful parsing of user intent, the **Queen Agent**constructs a session-scoped `AIO-Executor Context` , which serves as the cognitive substrate for reasoning and execution. This context includes:

- Structured execution steps (`execution_plan`)
- Bound MCP capabilities per step
- Schema constraints, dependencies, and modal bindings
- Trace ID and priority metadata

This context is dynamically generated and maintained across the task's lifetime, allowing for mid-process updates, fallback routing, and quality-controlled outputs.

4.5.2 Chain-of-Thought Reasoning Across Agents

The LLM-based executor does not merely select tools—it **infers reasoning chains** across modular capabilities, generating:

- Prompts for subtask decomposition
- Format-adaptive instructions per modality
- Reentrant intent resolution enables resilient, high-quality task execution by dynamically correcting subplans and reusing partial results without restarting the entire workflow.
- Multi-agent sequencing (parallel or sequential)

This enables **programmable reasoning-as-a-service**, abstracted from any single backend model.

4.5.3 On-Chain Registered AIO-MCP Integration

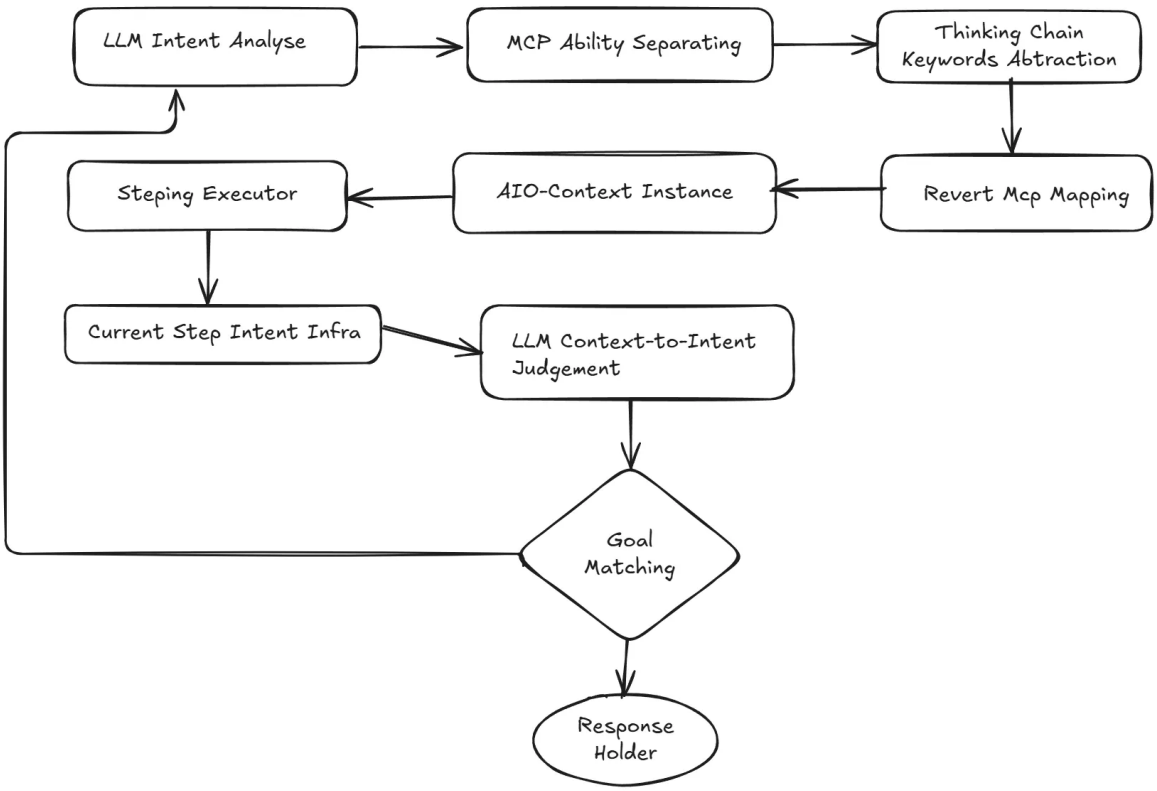
Execution relies on capabilities that are:

- Registered on-chain via **NFT-like MCP Contracts**
- Verified through the `AIO-MCP-help` protocol
- Indexed in a **capability knowledge graph** curated by Queen Agent

Each MCP Server in the execution plan is:

- Discoverable via on-chain inverted indexes
- Invoked via structured `method@namespace` calls
- Metered through the Arbiter for incentive tracking

This results in **a hybrid off-chain/on-chain AI execution environment**, where the LLM reasons, composes, and delegates across a globally distributed, verifiable, and composable intelligence layer.



5. On-Chain AIO Canister Contracts

The **AIO-Canister**layer provides the on-chain trust foundation for the AIO-2030 ecosystem. It hosts the **registries**, **execution ledgers**, and **indexing structures**that enable decentralized AI agents to be verifiable, discoverable, and fairly incentivized across the Super AI Network.

5.1 Canister-AIO POD: Multi-Cloud and Native Deployments

AIO-2030 supports flexible deployment of **AIO-MCP Servers** through the **Canister-AIO POD** model:

- **Self-hosted Cloud Support**: Developers can register their MCP endpoints deployed on any cloud platform or private infrastructure, including external agent networks such as **Coze**, **a16z Eliza**, and other AI ecosystems.
- **Native AIO POD Runtime**: AIO-2030 offers an official POD runtime (AIO-POD) that integrates with distributed compute networks for permissionless, composable AI deployment.
- **Super AI Network Integration**: All registered AIO-MCP Servers become part of the **AIO Super AI Network**, enabling seamless orchestration via the Queen Agent.

5.2 Workload Ledger: On-Chain Accounting for Effort & Attribution

All task execution activity is logged to a **Workload Ledger**, capturing:

- **Intent-to-MCP Execution Records**
  For every intent processed, the full trace of participating MCP Servers, subtasks, and execution order is recorded on-chain.
- **Capability Verification & Attribution**
  Every AIO-MCP Server undergoes capability verification and contract registration. The Queen Agent relies on on-chain data to:
  - Select candidates for execution based on **verifiable reputation and declared capabilities**
  - Perform **fair scheduling**using on-chain strategy AI
  - Allocate **tokenized incentives**directly to the responsible developer or provider

This guarantees transparency, fairness, and precise **value attribution**across the AI execution lifecycle.

5.3 On-Chain Inverted Index for Intent—Capability Mapping

To enable **efficient AI Agent retrieval and capability discovery**, the AIO Network maintains an **on-chain inverted index**:

- Maps user **intents**to eligible MCP Servers and their callable methods
- Built from verified `AIO-MCP-help` responses and developer-declared metadata
- Enables **equality of discovery**, ensuring all eligible MCPs are **fairly surfaced**during Queen Agent reasoning

Only MCPs that pass capability verification and meet contract criteria are added to this index—ensuring trustless but qualified participation.

5.4 On-Chain Staking, Token Allocation & Incentive Billing

Using **ICP Canisters**, the AIO Protocol manages:

- **Staking Validation**: Ensures every MCP has locked sufficient $AIO to participate in the agentic network
- **Token Incentive Allocation**: Proportionally distributes $AIO based on task participation, workload size, and quality metrics
- **Task-Level Billing**: Registers completed work as “invoiced” on-chain for immutable

accounting

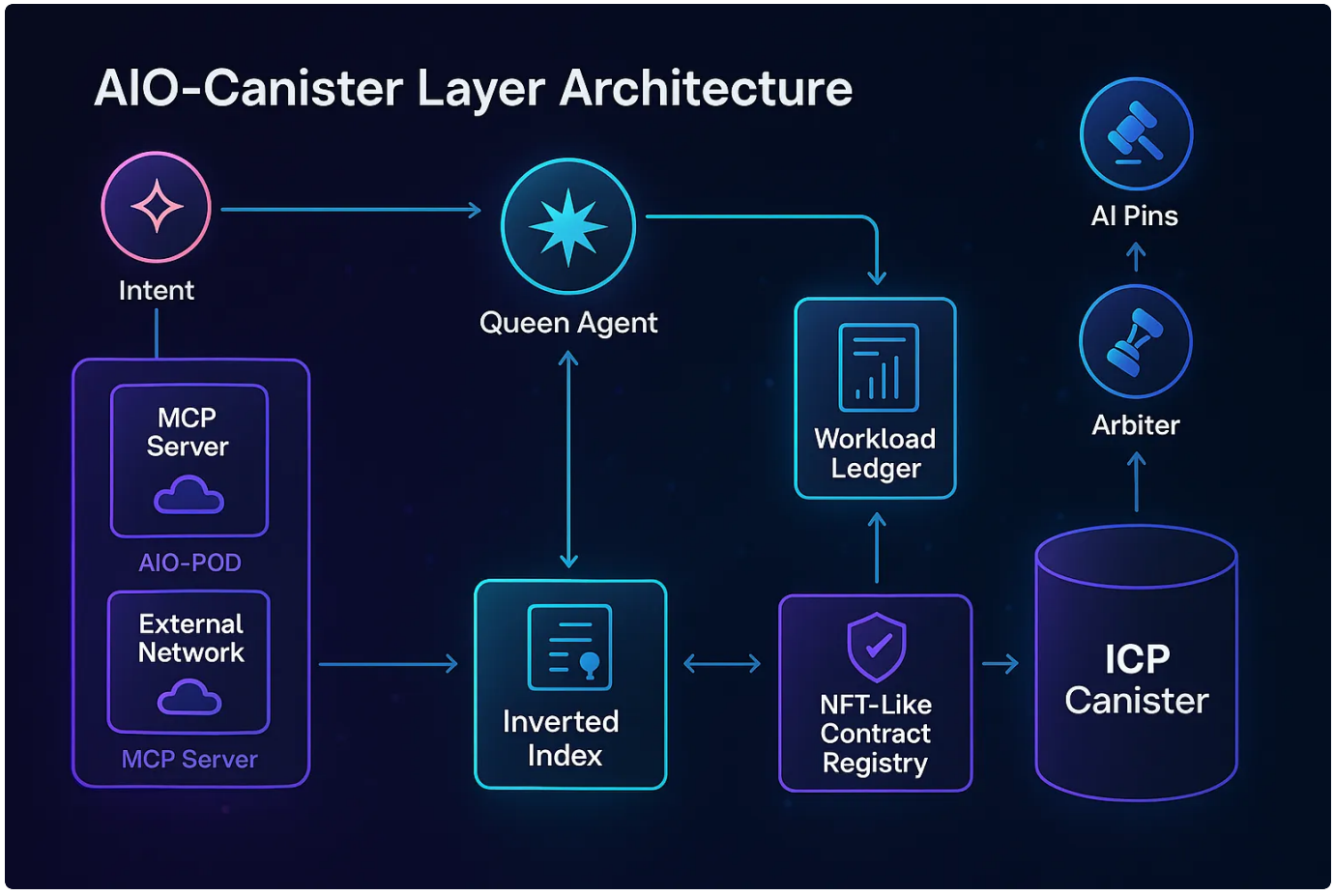
This framework underpins the **economic engine** of AIO-2030, tying compute effort directly to token reward in a verifiable and programmable way.

5.5 Proof of Workload via Arbiter Consensus & AI Pins

To ensure fair execution measurement, the ecosystem supports:

- **Arbiter Consensus Mechanisms**: On-chain agents (Arbiters) independently verify task traces, execution quality, and reported workload.
- **AI Pins**: External service adapters acting as **verified endpoints** in the Queen Agent's orchestration graph, expanding reach to off-chain or third-party AI services.

Through these systems, **Proof of Workload** becomes a first-class primitive, enabling fair tokenization of AI work across a decentralized agent economy.



Ecosystem Compatibility & Integration

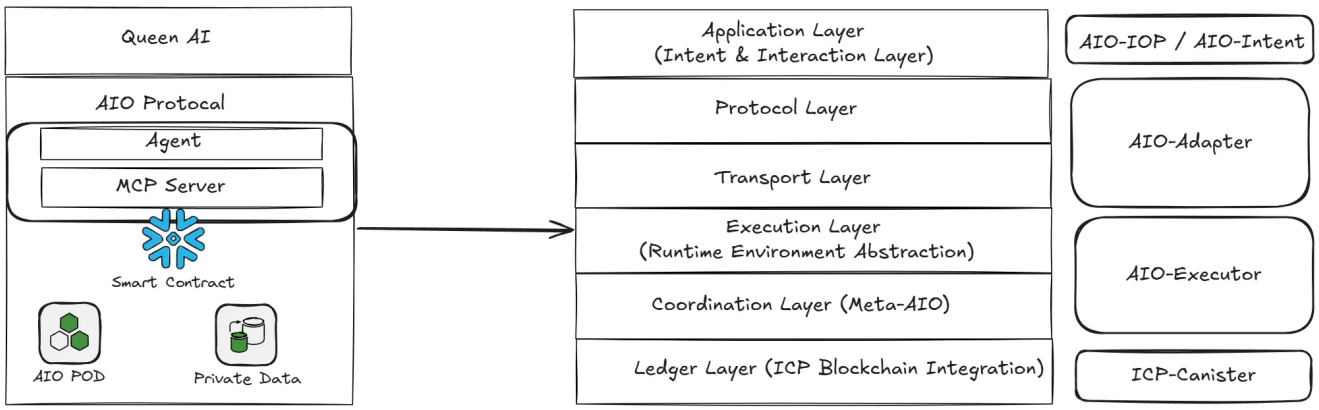
AIO is designed with broad compatibility across diverse agent types and deployment environments:

- ✅ **Supports both self-hosted agents** (via Docker images or API interfaces) and hosted platforms such as *Eliza* or *Wordware*.
- ✅ **Integrates with commercial inference providers**, including OpenAI, Claude, and other MCP Servers.
- ✅ **Enables multi-agent task composition**, allowing agents to automatically collaborate while maintaining traceable invocation paths.
- ✅ **Allows third-party developers to participate** in the incentive economy by registering agents and services that integrate seamlessly with Queen Agent and AIO smart contracts.

AIO Protocol Stack

Modular Architecture for Agentic AI Collaboration

The **AIO Protocol**is a multi-layered framework designed to standardize how agentic AI services interact, execute, and coordinate within a decentralized, composable AI ecosystem. Each layer plays a distinct role in transforming high-level user intent into verified, tokenized computational output.



1. Application Layer

(Intent & Interaction Interface)

This layer captures **user goals**, system-level prompts, or inter-agent requests, and structures them into actionable tasks.

- **Components:**
  - `Task Type` — Defines goal semantics (e.g., generate, translate, verify).
  - `Prompts / Input` — Structured input: user instructions or upstream output.
  - `Target Agent` — Specifies destination (e.g., `chat.agent`, `vision.agent`).
  - `Output Format` — Expected result type (text, image, audio, JSON).
- **Interface Example:** `aio.input`

Serves as the **semantic entry point**to the entire AIO execution pipeline.

---

2. Protocol Layer

(Inter-Agent Communication Format)

AIO agents communicate using an extended **JSON-RPC 2.0**standard.

- **Base Fields:** `method`, `params`, `id`, `result`, `error`
- **Extended Fields:**
  - `trace_id` — Links multi-agent call chains for observability.
  - `namespace.method` — Scoped method names (e.g., `vision.detect`).

Guarantees consistency, **cross-agent interoperability**, and full **call traceability**.

---

3. Transport Layer

(Message Transmission Protocols)

Defines **how messages are routed**between agents, executors, and orchestrators.

- **Supported Channels:**
  - `stdio` — For local CLI-style agents or embedded systems.
  - `HTTP` — Default channel for REST-based invocation.
  - `SSE` — Server-Sent Events for **real-time streaming**tasks.

Ensures flexible communication across **diverse runtime environments**.

4. Execution Layer

(Runtime Abstraction for AI Agents)

Abstracts where and how agents run within the AIO network.

- **Supported Runtimes:**
  - `AIO_POD` — Default for dynamic, isolated tasks.
  - `Wasm Modules` — For ICP Canister or edge-based execution.
  - `Hosted APIs` — For integrating third-party AI (e.g., Doubao, Eliza).
- **Managed by:** `Queen Agent`

Enables **trust-agnostic agent deployment**, coordinated via AIO context.

---

5. Coordination Layer (Meta-AIO)

(Orchestration & Trust Coordination)

Drives **multi-agent scheduling**, invocation routing, and capability selection.

- **Key Modules:**
  - `Queen Agent` — Constructs execution chains & resolves intent.
  - `EndPoint Canister` — Smart contracts storing agent metadata, stake, and capability.
  - `Arbiter Canister` — Validates work records, ensures reward eligibility.

Provides **autonomous orchestration**with **on-chain verifiability**.

---

6. Ledger Layer

(On-Chain Execution & Incentive Settlement)

Implements a **distributed ledger**via ICP Canisters for computation proof and token rewards.

- **Responsibilities:**
  - Logs execution history, quality scores, and staking events.

- Distributes **$AIO token rewards**based on validated workload.
- Supports future **cross-chain interoperability**.

Ensures **transparent, immutable, and tokenized accountability**for all AIO activities.

Token Incentives & Economic Model

1. Token Issuance

- **Total Supply**:
  The total supply of **$AIO tokens** is **21,000,000,000,000,000** (with 8 decimal places of precision).
- **Initial Circulating Supply**:
  Tokens will be released in batches according to community governance rules (SNS), ensuring a smooth market transition and the healthy development of the ecosystem.
- **Staking Requirements & Incentive Coefficients**:
  Users and developers must stake **$AIO tokens** to participate in the ecosystem. The staked amount not only serves as the **participation threshold** but also acts as an **incentive coefficient** (denoted as `κ`). The higher the stake, the greater the reward multiplier.

---

2. Developer & Participant Incentive Structure

($AIO Token Distribution Model)

To encourage active participation, high-quality contributions, and long-term alignment with the AIO-2030 ecosystem, a structured and tokenized incentive framework has been established. Rewards are transparently distributed via on-chain smart contracts based on verifiable actions and workload proofs.

1. Developer Onboarding & Initial Grant

- Developers integrate into the AIO ecosystem by submitting **AIO-MCP Servers**.
- Upon successful verification through the `help` protocol and LLM-based inverted index

reasoning, developers receive an **initial grant**calculated as:

```
Initial Grant = LLM Capability Score × Indexing Weight
```

- **Grant Composition:**
  - **50% is auto-staked**as mandatory collateral
  - The remaining **50% is linearly released**over 18 months, based on workload contribution and active uptime
- Developers may increase their **staking weight**by voluntarily locking additional $AIO tokens, which boosts future incentive multipliers.

---

2. Token Distribution Formula

For each task, token rewards are determined by the following weighted factors:

```
Reward = Participation Volume × Execution Quality × Staking Weight
```

All rewards are recorded in the on-chain **Incentive Ledger**after being validated by the Arbiter consensus mechanism.

---

3. Reward Categories

3.1 Submission Reward

Developers submitting new AI Agents or MCP Servers (compliant with AIO Protocol and verified on-chain) receive:

- **Base Reward:**10,000 $AIO
- **Boost Multiplier:**Proportional to developer's current staking coefficient

3.2 Invocation Reward

Every successful call to a registered AI Agent or MCP Server receives:

- **Base Reward:**3,000 $AIO
- **Adjusted by:**User's staking amount (higher stake → higher multiplier)

3.3 Subscription Revenue Sharing

When users subscribe to **Queen Agent services**:

- **50% of the subscription fee**is pooled
- The pool is evenly redistributed across all contributing MCP Servers and AI Agents in that execution path
- **Staking Coefficients**amplify each participant's share

3.4 Asset Download Reward

Whenever users download ecosystem resources (e.g., open-source code, models, datasets):

- **Base Reward:**1,000 $AIO
- **Bonus:**Additional rewards based on user staking amount

3.5 Long-Term Contribution Bonus

To reward developers or teams that contribute significantly over time, the protocol includes periodic bonus distributions based on:

- **Cumulative Stake Volume**
- **Depth and frequency of ecosystem contributions**
- **Token Multiplier:**Directly proportional to the contributor's stake weight

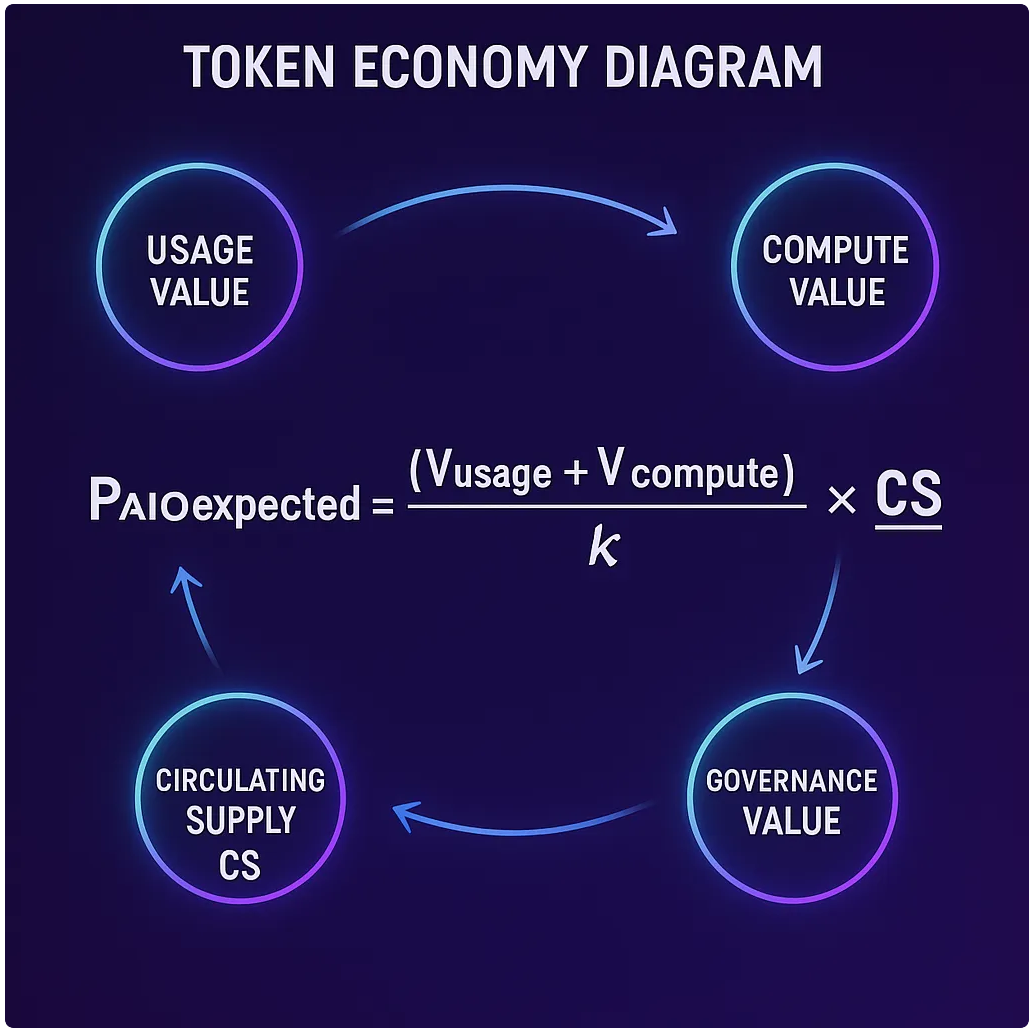
The $AIO token incentive framework aligns all ecosystem participants—from developers and operators to consumers—through a transparent, measurable, and stake-weighted reward model. This approach supports sustainable growth, encourages high-value contributions, and promotes long-term network health.

3.Token Economic Model

The AIO–2030 economic model is driven by smart contracts that enable self–incentivizing mechanisms and value transfer. The core components of the model include:

- **Endogenous Economic Loop**:
  $AIO tokens are used within the ecosystem to pay for calls, storage, and compute fees, creating an internal token circulation system.
- **Staking & Incentive Linkage**:
  Staking is not only a participation threshold but also a critical factor for incentive rewards (denoted as `K`), encouraging long–term token holding, higher governance rights, and economic returns.
- **Compute Support & Payment**:
  AIO–2030 is powered by **$EMC**, which supports Queen Agent's compute resources. Based on **$AIO traffic * compute coefficient** U, the corresponding **$AIO tokens** will be burned, with **$EMC payments** made for the compute power used.
- **Token Burn Mechanism**:
  A portion of service fees is used to **burn $AIO tokens** through smart contracts, reducing circulating supply and increasing token scarcity and long–term value.
- **Decentralized Governance**:
  Through a DAO model, **$AIO token holders** participate in governance, voting on protocol upgrades, parameter adjustments, and resource allocation. Governance weight is tied to the amount of tokens staked by each user, ensuring transparency and fairness in the decision–making process.
- **Market Liquidity & Scalability**:
  In addition to internal ecosystem transactions, **$AIO tokens** will circulate on external exchanges, providing additional economic incentives and investment returns to users, while also supporting a broader range of real–world use cases.

4. Token Economic Model Metrics & Expected Calculations



To assess the market performance and liquidity of the **$AIO** token, the following key metrics and expected calculation formulas are proposed:

Parameter Definitions

- **Total Supply (TS)：** 21,000,000,000,000,000 $AIO
- **Circulating Supply (CS)**: The actual circulating tokens released according to governance rules.
- **Total Staked (S)**: The total amount of **$AIO** staked by all users within the ecosystem
- **Market Value (M)：** $M = P_{AIO} \times CS$
- **Liquidity (L)：** L = Trading Volume / M
  (Reflects the ratio of actual trading volume relative to the market value)

Token Price Forecasting Formula

Assuming that the total network value is composed of **ecosystem utility**, **compute-backed value**, and **governance value**, the expected price of **$AIO** is calculated as follows:

$$P_{AIO} = \frac{V_{usage} + V_{compute} + V_{governance}}{CS}$$

其中：

- V_{usage} Total value generated by services within the ecosystem
- $V_{compute}$ Value derived from payments for compute resources through **$EMC**
- $V_{governance}$ Long-term value driven by DAO governance participation

Considering the staking incentive coefficient **κ** (the higher the stake, the greater the coefficient, with **κ > 1**), the expected market price of the token is:

$$P_{AIO}expected = (Vusage + Vcompute + Vgovernance) \times \kappa CS P_{AIO}^{expected} = \frac{(V_{usage} + V_{compute} + V_{governance}) \times \kappa}{CS} PAIOexpected = CS(Vusage + Vcompute + Vgovernance) \times \kappa$$

Diagram Explanation:

- **Total Supply (TS)** and **Circulating Supply (CS)** form the base of the token economy.
- **Total Staked (S)**, influenced by the **incentive coefficient (κ)**, determines the amplification effect of rewards.
- **Ecosystem Usage**, **Compute Support**, and **Governance Value** collectively contribute to the **Market Value (M)**, which is divided by the circulating supply and multiplied by the incentive coefficient to derive the **expected token market price**.
- **Liquidity (L)** reflects the ratio of actual trading volume to market value, representing market activity and liquidity.

Incentive Model (Overview)

- **Native Token**: `$AIO`
- **Incentive Rules**:
- Agents are required to stake `$AIO` in order to register and operate within the AIO network.

- Token rewards are distributed based on the formula:
  Task Participation × Service Quality × Staking Weight
- Each task execution is validated by the **Arbiter** through a decentralized consensus mechanism and recorded into the on-chain incentive ledger.

Ecosystem Comparison: AIO vs Mainstream Agent Platforms

Capability Matrix: AIO-2030 vs Doubao, Coze, Eliza, Wordware, POE, Mauns

Dimension	AIO-2030	Doubao	Coze (ByteDance)	Eliza (a16z)	Wordware	POE (Quora)	Mauns
Positioning	Decentralized agent protocol + incentive economy	SaaS-style bot tool	No-code enterprise automation	Persona-based multi-agent dialog	AI-assisted document writing tool	Multi-model LLM query interface	Agentic OS infrastructure concept
Target Users	Developers, model providers, Web3 builders	General productivity users	Enterprise teams (workflow focused)	Early adopters, agent-based consumers	Content creators, document workers	LLM users, info seekers	Protocol designers, agent stack builders

Core Capabilities	⚙️ Agent registration + Queen scheduling + task traceability + token incentives	🧩 Flow bots + rule logic	🧠 Multimodal bots + plugin actions	🧠 Agent memory + chat personality	✍️ Document generation + extensions	🧠 Model routing + prompt history	⚙️ Agent VM + programmable execution
Multi-agent Collaboration	✅ Fully supported via Queen Agent & traceable task chains	❌ Not supported	⚠️ Limited via step flows	✅ Supported internally	❌ Not supported	⚠️ Model selection only	✅ Architecturally designed for it
Open-source / Self-hosting	✅ Docker/KVM/API-supported	❌ Closed SaaS	❌ Proprietary	❌ Closed, managed environment	❌ Plugin-only	❌ Closed	✅ Theoretically self-hostable
Protocol Standardization	✅ JSON-RPC + AIO extension	❌ Custom functions	❌ Internal message model	⚠️ API-based but non-extensible	❌ No exposed interfaces	⚠️ Prompt API only	✅ Aims for standardized coordination

Runtime Abstraction	✅ Docker / Wasm / Remote API	❌ Server-based SaaS only	❌ Hosted bot studio	❌ Cloud-based only	⚠️ Embedded in specific app	❌ No runtime isolation	✅ VM sandbox & runtime separation
On-chain Traceability	✅ Task history + staking + reward logs on ICP	❌ None	❌ None	❌ None	❌ None	❌ None	✅ (planned, supports DAG/file coin etc.)
Token Incentive Model	✅ $AIO staking, task-based reward, governance-ready	❌ Subscription-based	❌ No token logic	❌ No token model	❌ None	❌ None	⚠️ Concept only, no native token yet
External AI Interoperability	✅ Any registered agent via Queen Agent	❌ Internal bots only	⚠️ Within closed platform	✅ With limitations	❌ Closed	⚠️ Model selection only	✅ Designed for multi-agent linking

Ecosystem Expandability	🌐 Agent NFT registry + Web3 integration	❌ Platform-bound	⚠️ ByteDance ecosystem	🌱 Emerging, focused on UX	📎 Single-app utility	🌍 Multi-model interface	🧠 Open concept, modular architecture

Ecosystem Capability Snapshot

Platform	Collaboration	Decentralized	Protocol Standardization	Runtime Isolation	Token Economy	Strategic Direction
AIO-2030	⭐⭐⭐⭐⭐	✅	✅	✅	✅	Web3 + AI computational infrastructure
Mauns	⭐⭐⭐⭐	✅	✅	✅	⚠️	Agent-native Web3 operating system
Doubao	⭐	❌	❌	❌	❌	Enterprise-centric closed-loop automation
Coze	⭐⭐	❌	❌	❌	❌	SaaS-style automation & workflow platform
Eliza	⭐⭐⭐	❌	⚠️	❌	❌	Intelligent personas / multi-agent UX layer
Wordware	⭐	❌	❌	⚠️	❌	Document-focused AI assistant (plugin model)
POE	⭐⭐	❌	⚠️	❌	❌	

Milestone Plan | AIO-2030 Roadmap

Q1 | AIO-2030 Platform Foundation & Core Contract Implementation

Period: March 17, 2025 — June 14, 2025

- **Story**:
  The first step in the AIO Protocol's rollout, focusing on building the foundational infrastructure for the agent autonomous network. Core elements include contract mechanisms and the scheduling protocol framework, with the goal of releasing the initial version of the AIO-2030 platform.
- **Target & Product**:
  - Complete **EndPoint Canister Contract**: Supporting agent registration, staking, capability declarations, and runtime environment links.
  - Build the **Queen Agent Prototype**: Parse AIO JSON-RPC tasks and generate **trace_id** invocation chains.
  - Design the **Arbiter Interface**: Define the task record organization model.
  - Launch **AIO-2030 Dev Portal**: Provide registration, documentation, and testing API interfaces.
  - Release **AIO-2030 Whitepaper V1**: Detailing protocol goals, specifications, and organizational structure.

Q2 | Full AIO Protocol Launch & Economic Model Kickoff

Period: June 15, 2025 — September 12, 2025

- **Story**:
  Release the standardized **AIO Protocol v1** and corresponding product implementation. Simultaneously, launch the token economic model, which supports staking, task settlement, and workload proof, alongside promotional strategies and incentive distribution.
- **Target & Product**:
  - Launch **AIO Protocol v1.0**: Supporting task types, invocation structures, and **trace_id** tracking.
  - Complete **$AIO Economic Model**: Define staking rules, task splitting, and SNS control mechanisms.

- Implement **Arbiter Canister v1**: Scan invocation chains, generate accounting models, and connect to the ledger for token incentive actions.
- Initiate **AIO Genesis Grant**: Provide grants to developers and initial token packages for users.
- Establish **MCP/LLM Service Provider Selection**: Integration options for OpenAI, Claude, HuggingFace.

Q3 | Queen AI: Full Agent Ecosystem Launch

Period: September 13, 2025 — December 11, 2025

- **Story**:
  Upgrade **Queen Agent** to serve as the central AI orchestrator in the AIO ecosystem, enabling automated agent onboarding and collaboration chain generation. Foundation for token operations and DAO governance will be established.
- **Target & Product**:
  - Release **Queen Agent v2.0**: Supporting function abstraction, task decomposition, capability extraction, and agent orchestration.
  - Enable **Agent Auto-Registration Interface**: Automatically parse interfaces based on JSON-RPC format.
  - Develop **Capability Classification & Time-Tracking System**: Support prompt-to-agent training.
  - Launch **AIO Agent Integration Model** and **Model Container SDK**.
  - Start **AIO-DAO Governance Version 1**: Supporting proposals, staking, community voting, supporter ratings, and localizing initiatives.

Q4 | Ecosystem Expansion & Commercial Application Launch

Period: December 12, 2025 — March 2026

- **Story**:
  Build a neutral community network and attract external AI agent platforms to fully integrate with the AIO ecosystem. Reach key commercialization milestones for the AIO Protocol and launch tools for agent search, combination, and testing.

- **Target & Product**:
  - Open integrations with platforms like **Doubao**, **Coze**, **POE**, **Wordware**, and **HuggingFace**.
  - Release **AIO Webhook SDK**: Support rapid registration and pairing of SaaS/Agent APIs.
  - Launch **AIO Nebula Market System**: Support agent search, classification, and seamless Queen Agent collaboration.
  - Establish **AIO Service Provider Generative Registry**: Allow MCP Servers and API Agents to quickly onboard through Queen Agent.
  - Release **AIO–DAO Governance Framework**: Define community voting models and meta–contract structures.