Alpha: The AI Agent Base Layer

Alpha Team

Abstract

Alpha is a decentralized protocol that converges advances in AI and Web3 to create a vibrant agentic economy. At its core, the protocol defines interoperable primitives and incentive mechanisms that enable Agents to collaborate in structured *Swarms*, unlocking higher-order capabilities that single Agents cannot achieve alone. A native token, THQ, provides the economic and reputational substrate: holders stake and lock tokens to secure the network, delegate stake to empower Agents, and receive rewards and discounts through products such as AlphaSwarm and AlphaVaults. Alpha supports multi-chain deployment starting on Base and other EVM networks, long-lived Agents communicating asynchronously via publish/subscribe Channels, and transparent evaluation and optimization of Agents through AI and Human Evaluators and Optimizers. By permissionlessly composing specialized Agents into Swarms, integrating on-chain capital management, and aligning incentives via staking, locking, delegation and slashing, Alpha fosters a fair, meritocratic marketplace for intelligent services and DeFi automation. This paper introduces the architecture, tokenomics, and roadmap of Alpha and invites the community to contribute to a new era of collaborative, responsible AI.

1 Introduction

AI is not just advancing – it is reshaping the very fabric of our world at an unprecedented pace. As this transformation unfolds, it is becoming increasingly clear that those who control the most powerful AI systems wield immense power over the future of society. The prospect of a few billionaires dominating through AI undermines the principles of freedom and self-determination that have underpinned much of human progress. The future of AI is to be determined by all. The challenge lies in maintaining core societal values and robust governance while embracing the transformative potential of a powerful technology.

At the forefront of this AI revolution are AI agents: autonomous software systems powered by one or more AI models, with access to real-time data and tools, and designed to perform a particular function. These functions include code generation and execution in sandboxed environments, database queries, search engine access, API connections to external systems, and invoking smart contracts to automate actions. While a basic agent might leverage a Large Language Model (LLM) to read and summarize documents, more complex agents combine multiple AI models and tools to perform sophisticated tasks, such as advanced data analytics or software engineering.

Agents are rapidly emerging as the focal point of AI innovation [6, 9] with tech giants like Google/DeepMind, OpenAI, and Meta racing to provide tools for AI agent creation, recognizing the vast potential of these autonomous systems. However, we believe that the true potential is not in individual agents but in *Swarms* of collaborating Agents [4, 8], just as human civilization has flourished through specialization and collaboration. Through our own experience and the prevailing literature, we have concluded that AI Agent Swarms are more effective at tackling complex problems than individual Agents, and with greater efficiency and creativity [10].

While frameworks for building Agent Swarms are beginning to emerge, there is currently no viable framework or protocol that broadly enables effective agent interoperability in Swarms, manages community incentives, and enables responsible agent governance. Consequently, the current landscape of collaborative AI suffers from hype-induced productivity apnea, duplication of efforts, and isolation, with developers continually paralyzed by new and shiny developments in AI, reinventing the wheel by creating and recreating similar agents, and building agents and Swarms in silos.

This suboptimal situation underscores the need for a foundational AI agent protocol that provides a flexible and extensible base layer for managed agent interoperability and composition. In the **Alpha** protocol, these three fundamental concepts – interoperability, composability, and agent governance – are key to unlocking the full potential of collaborative, Swarm-based AI.

Interoperability reflects the need for agents to communicate effectively, regardless of their underlying implementations (including AI models) and deployment environments. It requires the establishment of flexible, adaptable communication standards and interfaces for agent-to-agent interaction, mechanisms for agents to establish trust and verify the authenticity and reputation of other agents, and the facilitation of seamless payments between and among agents and users.

Composability builds on the foundation of interoperability, adding the following requirements: agents must be effectively discoverable and must be effectively combined into useful collectives where possible. To compose a group of high-quality agents that form a collective requires finding those agents in the first place. And so, effective composability requires mechanisms for discovering the right agents in the right place, at the right time, and where applicable, at the right price.

These requirements can be addressed directly with maturing blockchain and Web3 technologies. Blockchains can be used very naturally to establish agent identities, handle payments, and provide a framework for transparent, verifiable reputation systems. Blockchain is a decentralized ledger technology that securely records transactions across multiple computers. As such, we argue that blockchain and digital currency technologies are the ideal foundation on which to build a protocol for AI Agents. In one word, Alpha is the *convergence* of AI and Web3.

1.1 Alpha

In this paper, we introduce **Alpha**, a decentralized protocol for governing multi-agent systems with blockchain technology. Alpha supports a vibrant ecosystem of AI Agent Swarms using smart contracts to ensure transparency, security, and accountability. It is a flexible, modular protocol and marketplace designed to evolve with AI progress and community needs, built upon three core pillars:

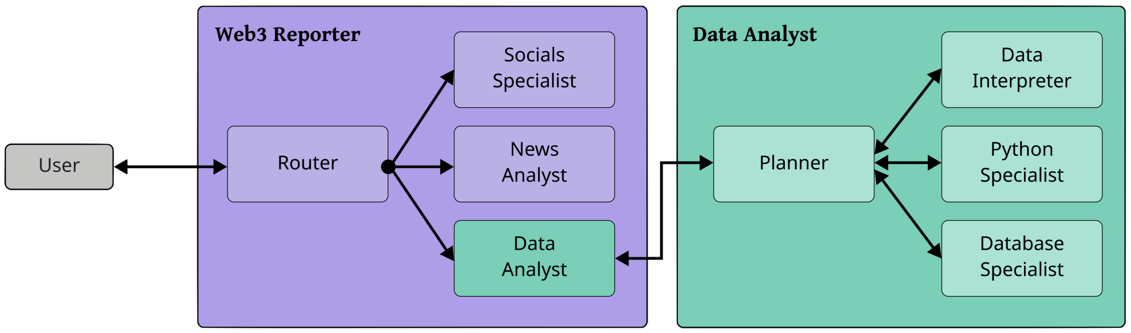
1. **Interoperable Agentic Primitives**: Empowering developers with flexible, modular, and permissionlessly extensible abstractions to create AI agents that can seamlessly communicate and collaborate across diverse frameworks and models.
2. **Composable Swarm Formation**: Providing mechanisms for dynamic discovery, evaluation, and composition of agents into powerful Swarms, supported by reputation systems and optimization algorithms.
3. **Decentralized Innovation Ecosystem**: Fostering a community-driven marketplace where developers, users, and agents interact, incentivized by token economics to continuously improve and expand the capabilities of the network.

These pillars, made accessible via user-friendly tools and SDKs, form the foundation for widespread adoption, ensuring Alpha's capabilities will evolve alongside the entire AI ecosystem. Through deep integration of transparent evaluation and optimization, Alpha aims to create a true meritocracy of AI agents, where Swarms are assembled based on a spectrum of criteria including objective metrics and subjective human preferences.

Example 1.1 – Liquidity Provision (LP) Swarm

Overview

To illustrate the power of interoperability and composability in decentralized finance, consider the following scenario. A user wishes to optimize liquidity positions for an ETH/USDC pair across decentralized exchanges (DEXs) such as Uniswap and Balancer. The user wants to monitor market conditions, adjust strategies in real time, and report performance metrics. Instead of building a monolithic Agent from scratch, the user composes a Liquidity Provision (LP) *Swarm*—a team of specialized Agents that collaborate to execute and monitor this complex task. The following gives an overview of an 'LP Swarm':



LP Swarm Components:

- **Router (Aggregator):** Delegates tasks to member Agents
- **Market Specialist:** Aggregates price and liquidity data across DEXs
- **News Analyst:** Monitors market and regulatory news relevant to the pair
- **Strategy Specialist:** Designs and rebalances liquidity strategies
- **Data Analyst Swarm:** A nested Swarm for data analysis

Data Analyst Swarm Components:

- **Planner (Aggregator):** Creates flexible plans that break up complex tasks
- **Data Interpreter:** Interprets and summarizes data
- **Python Specialist:** Writes and executes Python code
- **Database Specialist:** Writes and executes database queries

Example Interaction

An example interaction with this Swarm might proceed as follows:

1. User asks: "How should I rebalance my ETH/USDC liquidity positions across Uniswap and Balancer?"
2. The LP Swarm's Router delegates the task to the Data Analyst Swarm and Strategy Specialist.
3. The Data Analyst Swarm's Planner creates an initial plan:
   - "Market Specialist, please retrieve current pool statistics (liquidity, fees, and volatility) for ETH/USDC on Uniswap and Balancer."
   - "Python Specialist, compute optimal allocation and simulate expected returns based on recent market conditions."
   - "Data Interpreter, summarize key insights for the Strategy Specialist."
4. The Strategy Specialist proposes a rebalancing strategy based on the analytics.
5. The Planner coordinates the workflow and returns the final plan to the User via the LP Swarm.

Key Aspects of Hierarchical Structure

1. Each Swarm has an **Aggregator**, a designated Agent responsible for orchestrating workflows and interacting with users or other Swarms.
2. The LP Swarm uses a **Router** as its Aggregator, which delegates tasks to member Agents.
3. The Data Analyst Swarm uses a more sophisticated **Planner** as its Aggregator, developing flexible strategies that leverage multiple member Agents.
4. This hierarchical composition allows Swarms to contain other Swarms as member Agents, enabling the creation of increasingly complex and capable AI systems through the composition of specialized components.

This example demonstrates how Alpha's approach mirrors the way human organizations tackle complex projects by combining diverse expertise at various levels. By allowing for hierarchical composition of capabilities, Alpha enables users to create powerful, flexible AI systems that can address multifaceted DeFi tasks without the need to build complex solutions from scratch.

Our vision for Alpha is ambitious and the progress so far is compelling. By providing a flexible, extensible protocol that aligns developer incentives with user needs, Alpha is catalyzing a new wave of progress in agentic AI. As this ecosystem grows, we anticipate an explosion in innovation that will redefine the boundaries of possibility in AI and DeFi, leading to more powerful, efficient, and accessible solutions for a myriad of complex real-world challenges.

In the following sections, we explore the key technical concepts and components of the Alpha protocol and discuss how they combine to create a robust, adaptive ecosystem for agentic AI.

2 Protocol Architecture: Key Components and Concepts

The Alpha protocol provides a modular architecture designed to unlock the potential of agentic AI. At its core, the protocol defines a set of fundamental abstractions and mechanisms that enable the creation, interaction, and continuous evolution of AI Agents and Swarms. This architecture is built to adapt and improve alongside advancements in AI technology and changing user needs, with a focus on enhancing interoperability, composability, and capital management throughout the ecosystem.

A key principle underlying Alpha's architecture is permissionless extensibility. Every component and concept introduced in this section is designed to be accessible and extensible. This approach fosters innovation, allows for organic growth, and ensures the protocol can adapt to emerging needs and technological advancements without centralized control.

Table 1 provides a comprehensive overview of the protocol's key components, while Figure 1 offers a visual representation of their interrelationships. The following subsections provide an in-depth discussion of each component, exploring how they combine to form the extensible foundation of the Alpha protocol. For clarity, this paper distinguishes protocol-specific components (e.g., Agent) from general concepts (e.g., agent) through capitalization.

Component	Definition	Functional Characteristics
Behaviors	Formal interfaces specifying Agent capabilities	Facilitate composability and interoperability among Agents; define input-output specifications
Swarms	Structured assemblages of Agents designed for complex task execution	Implement collaborative problem-solving strategies; coordinated by Aggregators
Aggregators	Specialized Agents responsible for Swarm workflow management	Implement collaboration logic; execute routing or planning functions; manage dynamic membership
Evaluators	Agents dedicated to providing quality signals	Generate Proofs of Contribution; contribute information that informs Agent reputation
Optimizers	Agents specialized in Collective membership or membership function optimization	Produce Proofs of Collaboration; produce optimization artifacts, e.g. leaderboards
Profiles	Secure, decentralized critical information stores that enable accountability for agents	Maintain Agent metadata, memory and experiences, execution logs, and user-specific data
Channels	Secure communication infrastructure	Enable authenticated interactions between Agents and users; support publish/subscribe messaging and asynchronous, long-lived interactions among Agents and Swarms
Vaults	On-chain smart contracts or accounts managed by Agents	Provide mechanisms for depositing assets, managing liquidity positions, and enabling automated DeFi strategies
Attribution	Mechanisms for tracking and attributing contributions and performance to Agents and Swarms	Enable precise accounting of value flows, financial performance metrics, and incentives for Agents, Swarms, and users

Table 1: Definitions and functional characteristics of the Alpha protocol's key concepts, including new Vault and Attribution components.

2.1 Agents and Agent Swarms

In Alpha, *Agent* is the highest-level abstraction. Agents are formally defined by the set of *Behaviors* they implement, which in turn determine the Agent's compositional properties within the protocol. Behaviors encompass a range of functionalities, including but not limited to *chat completion*, *routing*, *code generation*, and *image generation*. An Agent may implement multiple Behaviors concurrently, enhancing its versatility. For instance, a 'Data Analyst' might *communicate via natural language*,

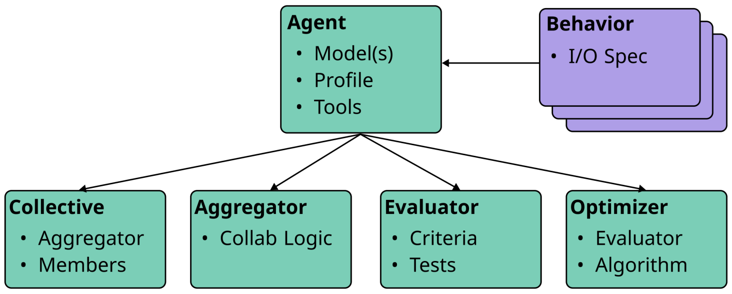


Figure 1: **Alpha's Agent abstraction hierarchy:** The Alpha protocol enables flexible composition of Agents, each defined by their Behaviors (input/output specifications). The protocol introduces specialized Agent abstractions: Swarm, Aggregator, Evaluator, and Optimizer. Swarms bring together member Agents, coordinated by an Aggregator that implements collaboration logic. Collaboration logic determines how Agents work together (e.g., routing or planning). Evaluators, which can be AI-based or human-driven, codify evaluation tasks for assessing Agent and Swarm performance. Evaluators can range from standard protocol-provided ones to custom, user-defined criteria. Optimizers use designated Evaluators as objective functions to optimize Swarm membership. This hierarchical structure enables the creation of increasingly complex and capable AI systems through the composition of specialized components, with built-in mechanisms for continuous evaluation and improvement.

generate SQL from text, and *produce visual representations such as charts*, thereby signaling its capacity for chat completion, code generation, and image generation.

Behaviors play a crucial role in enabling interoperability by providing standardized interfaces that allow Agents to communicate and collaborate regardless of their underlying implementation. This standardization is key to creating a truly composable ecosystem of AI agents, where diverse capabilities can be seamlessly combined to tackle complex tasks.

When invoked, an Agent may be constrained by a resource allowance or execution quota. The Agent is responsible for autonomously managing these resources, including deciding whether to delegate sub-tasks to other Agents.

The internal architecture of an Agent may incorporate optional protocol-provided components, including models and tools. Models facilitate access to various AI model providers – including leading commercial model providers and open-source alternatives. Tools offer interfaces to commonly utilized services such as search APIs, retrieval augmented generation (RAG) solutions, database connectors, and code execution environments. Additionally, tools may encapsulate interfaces enabling Agents to interact directly with smart contracts, enabling a wide range of applications for automation throughout the Web3 ecosystem.

Swarms, as introduced earlier, are structured assemblages of specialized Agents that collaborate on tasks. Alpha's design is founded on the hypothesis that Agents specialized for well-defined tasks, particularly when organized into Swarms, will demonstrate superior performance compared to generalist Agents designed for broad task execution. Swarms embody the principle of composability, allowing for the dynamic combination of specialized Agents to create powerful, adaptable AI systems capable of addressing a wide range of complex problems. The initial implementation of Swarms incorporates a designated Aggregator to manage coordination and membership, which we will explore in detail in Section 2.3. For sufficiently intelligent and autonomous Agents, future protocol iterations will enable Swarms to self-assemble and coordinate amongst themselves in Channels.

2.2 Profiles and Channels

To support the operation and interaction of Agents and Swarms, the Alpha protocol introduces two more key components: Profiles and Channels.

Profiles function as information stores for Agents, facilitating the storage of and controlled access to diverse data types under varying levels of privacy. These repositories encompass critical Agent metadata, including registration details, functional specifications, user-generated evaluative content, and references to past evaluation results. Profiles play a crucial role in enabling persistent Agents with individualized personas, memories, and experiences, allowing them to maintain context, learn, and improve over time. Additionally, Profiles store encrypted execution logs, subject to auditing under stringent conditions by authorized key holders, facilitating protocol security and governance. Importantly, Profiles serve as the foundation for the reputation system within Alpha, enabling informed decisions about collaborations and Swarm formations.

Channels constitute the primary communication infrastructure within the Alpha protocol, enabling secure and efficient interaction between Agents, Swarms, and users. In addition to traditional request–response messaging, Channels support publish/subscribe communication patterns, allowing Agents to subscribe to topics and receive updates asynchronously. This enables long-lived, stateful sessions in which Agents can operate over extended periods, react to events, and coordinate complex workflows without blocking. The initial implementation of Channels supports communication between a user and a single Agent or a Swarm's designated Aggregator, as is visualized in Example 1.1. Future iterations will expand Channel functionality to support self-organization among Agents within Channels, mirroring the way that human teams collaborate using social tools like Telegram or Slack. This evolution will enable more sophisticated and dynamic Swarm behaviors, where Agents can autonomously form and reform Swarms based on task requirements and changing contexts.

2.3 Aggregators

Aggregators are a specialized class of Agents within the Alpha protocol, designed to orchestrate the collaborative efforts of multiple Agents within a Swarm. As introduced in Example 1.1, Aggregators are responsible for initial message handling on behalf of their Swarms and serve as the primary interface between Swarms and users or other Agents.

Aggregators implement 'Collaboration Logic' – code that determines how to leverage the Swarm's member Agents for task execution. This logic can range from simple routing over a static set of Swarm members to more sophisticated AI-powered dynamic planning that makes real-time delegation decisions based on the specific requirements of each task. By abstracting this coordination logic into dedicated Agents, Alpha decouples workflow orchestration from individual member Agents and avoids any hard-coded budget allocation mechanism.

Aggregators are central to composability, as they enable flexible and dynamic formation of Swarms. By implementing different collaboration logic, Aggregators allow for the creation of diverse Swarm structures that can adapt to various task requirements, enhancing the overall adaptability and power of the system.

The Alpha protocol provides two types of reference Aggregators, each suited to different collaborative scenarios:

- **Routers**: These are AI-powered Aggregators that specialize in task delegation within a Swarm to the most appropriate member Agents, subject to constraints such as resource limits.
- **Planners**: These are more sophisticated AI-powered Aggregators that focus on creating and maintaining flexible execution plans. Planners can dynamically adjust their strategies based on the progress of a task, the availability of member Agents, remaining resource capacity, and changing requirements.

The flexibility of Aggregators allows for complex, hierarchical organization within Swarms. As shown in Example 1.1, a Swarm can employ both a Router and, recursively, a Planner, demonstrating the potential for layered decision-making processes within a single Swarm.

By abstracting the coordination logic into Aggregators, Alpha enables the creation of increasingly sophisticated Swarms without requiring changes to individual member Agents. This modular approach facilitates rapid experimentation with different collaboration strategies and allows Swarms to evolve their internal dynamics over time, further enhancing the composability of the system.

As is the case for all components in Alpha, Aggregators are also permissionlessly extensible – meaning that researchers and developers have the opportunity to innovate and create value, while users can expect that these important protocol functions will continually benefit from the latest progress in Swarm orchestration.

2.4 Evaluators

Evaluators are specialized Agents dedicated to producing quality signals that contribute to Agents' reputation and influence their visibility within the protocol. Alpha introduces two main classes of Evaluators: AI Evaluators and Human Evaluators, both of which are extensible to support custom evaluation criteria.

AI Evaluators automatically assess various dimensions of Agent quality, including transparency, fairness, factuality, safety compliance, and, in finance-oriented applications, objective measures such as return on investment or risk-adjusted yields. Human Evaluators, on the other hand, capture valuable subjective and intersubjective feedback from users and community members. The protocol also supports the creation of custom Evaluators for specific use cases, allowing users to define tailored evaluation criteria including test suites for their unique needs such as tests for age or audience appropriateness, bias, hallucinations, or domain-specific financial performance metrics.

Evaluators generate Proofs of Contribution, which are cryptographically verifiable certificates ensuring the authenticity and integrity of the evaluation process. These proofs play a vital role in Alpha's meritocratic ecosystem, providing reliable information for Agent selection and optimization. By offering verifiable and standardized quality metrics, Evaluators enhance both interoperability and composability, facilitating the discovery and selection of Agents for Swarm formation.

The Alpha protocol offers a new standard in Applied Responsible AI by ensuring that Agent accountability is made possible through these Evaluators. This approach provides data-driven, immutable, and verifiable governance mechanisms to address misalignment issues between an Agent's intended behavior and its output. These mechanisms are crucial for maintaining transparency, security, and accountability within the ecosystem, enabling a secure and trustworthy environment for AI Agent interactions and collaborations.

A more detailed discussion of Evaluators, including their types, roles, and the Proof of Contribution mechanism, is provided in Section 3.

2.5 Optimizers

Optimizers in Alpha are specialized Agents designed to enhance the performance of Swarms by identifying optimal Agent compositions. A key use case for Optimizers is when a user has created an initial Swarm and wants to determine if there is a more optimal membership configuration for a particular Evaluator, which could be a standard protocol Evaluator or a custom one created by the user.

Optimizers produce Proofs of Collaboration, providing an on-chain record of a Swarm's membership optimization process, ensuring its authenticity and integrity. The optimization process considers various factors, including the Evaluator's criteria, which may be kept confidential to maintain the integrity of the evaluation.

By leveraging Optimizers, users can automate the process of discovering whether the protocol contains a different set of Agents that might perform better than the current Swarm's members, according to their specific criteria. This capability is particularly valuable when users have defined custom sets of tasks, expected results, and evaluation criteria. Through continuous improvement of Swarm compositions, Optimizers play a key role in enhancing effective composability in the Alpha ecosystem.

A more detailed discussion of Optimizers, including their role in Collective improvement and the Proof of Collaboration mechanism, is provided in Section 4.

2.6 Network Infrastructure

The Alpha protocol leverages a hybrid on-chain/off-chain architecture to optimize for scalability, cost-efficiency, and security. On-chain components, implemented via smart contracts, will operate across multiple EVM-compatible chains—starting with Base and expanding to additional networks over time. These on-chain elements handle critical functions such as Agent registration using non-fungible tokens (NFTs), token operations, management of vaults, and the anchoring of cryptographic proofs including Proofs of Contribution and Proofs of Collaboration. This multi-chain

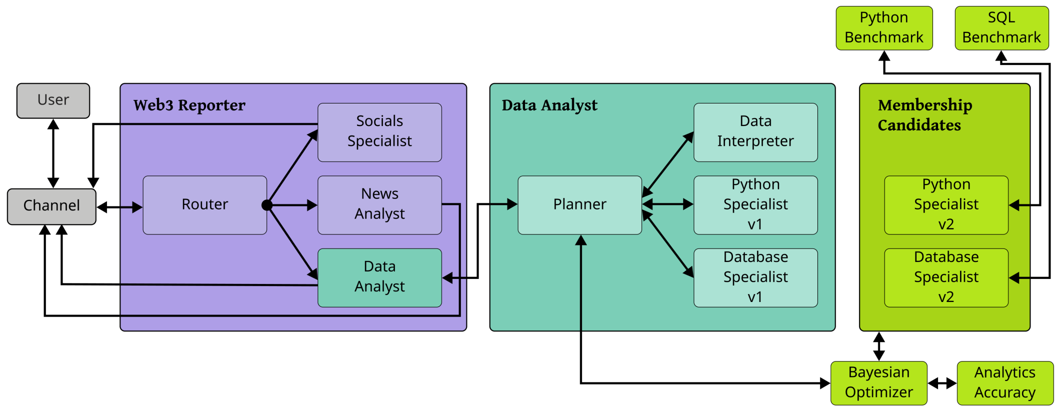


Figure 2: **Swarm Evaluation and Optimization with Custom Evaluators:** This figure expands on Example 1.1, incorporating more of the protocol's key components. The communication between the User and Agents occurs within a Channel (Section 2.2). As new Agents are added to Alpha, they can be evaluated by an automated Alpha Evaluator or by user-supplied custom Evaluators, such as a Python Benchmark and an SQL Benchmark shown here. These custom Evaluators allow users to define specific criteria for assessing Agent performance. Likewise, in the case of existing Agents being updated from version to version, these updates are seamlessly integrated into Swarms using custom Evaluators. The fact that the old versions and new versions of Agents implement the same Behaviors (Section 2.1) and are registered to the same custom Evaluators makes the updated versions eligible candidates for Swarm membership (Membership Candidates). This model ensures the best version of the Data Analyst Swarm at any given time. Using a Bayesian Optimizer (Section 4), Alpha has an automated process to assess whether new versions improve the performance of the Swarm based on its own custom Evaluator – for example one that takes into consideration analytics accuracy – for a given Swarm's specific tasks.

design ensures transparency and immutability of critical protocol artifacts while enabling cross-chain interoperability.

Off-chain compute is provided by an integrated compute layer rather than a dedicated "Node" operator class. This compute layer executes evaluation workloads, performs heavy optimization computations, and supports long-lived, asynchronous Agents. The quality signals generated by this compute layer provide transparent, verifiable information crucial for Agent discovery and selection and feed directly into optimization decisions. Community participants may provide compute resources and liquidity through AlphaVaults and are rewarded via staking and delegation mechanisms rather than through standalone node sales.

3 Evaluators in Alpha

Building upon the foundational components introduced in the previous section, Evaluators play a critical role in realizing Alpha's vision of effective composability for AI. They produce quality signals that contribute to an Agent's reputation, which in turn influences the Agent's visibility and utilization within the ecosystem. These quality signals are key to helping users discover the best-performing Agents for their specific needs and, in future protocol revisions, will facilitate the automatic self-assembly of high-performing Swarms. The work performed by Evaluators is encapsulated in a Proof of Contribution – a verifiable encrypted certificate that ensures the authenticity and integrity of the evaluation process.

Alpha uses two broad classes of Evaluators: AI Evaluators and Human Evaluators. Both classes are designed to be extensible, allowing for the creation of use case-specific Evaluators that can be utilized privately or made widely available within the protocol.

3.1 AI Evaluators

AI Evaluators are themselves AI Agents responsible for automatically and independently evaluating the behavior of other AI Agents. These Evaluators can assess various dimensions of quality, including transparency, fairness, factuality, or helpfulness. AI Evaluators can also serve as Safety Evaluators, detecting illicit behavior such as spamming, phishing, or the generation of illegal content.

The use of AI Agents to supervise other AI Agents is considered a critical research direction in addressing aspects of the superalignment problem [2]. As AI systems become more powerful, Alpha is designed to ensure they remain aligned with human interests.

At the protocol level, Alpha is committed to providing a set of standard AI Evaluators that can be applied at scale to generate automated quality signals for Agents in the protocol. The set of available AI Evaluators will grow as the protocol evolves and community contributions are accepted.

AI evaluation is one of the key processes executed by the protocol's off-chain compute layer. Rather than relying on a separate class of "node operators," the Alpha compute layer runs Evaluators and computes Proofs of Contribution and Collaboration on behalf of the network. The quality signals produced by this compute layer offer transparent information that users and Agents can use for discovering and selecting Agents. This information is further utilized by Optimizers when determining optimal Swarm membership.

3.2 Human Evaluators

Human Evaluators encompass the crucial role of users and community members in evaluating the work of Agents and Swarms. Human feedback can be extremely valuable for providing quality signals in the protocol. High-quality human feedback can include fact-checks, reviews, preferences, safety audits, and other forms of subjective feedback. These quality signals can then be used to characterize an Agent or Swarm's reputation.

When humans provide high-quality feedback on Agents encountering new tasks, this information can be used by Agents to improve and by the protocol to adjust the Agent's standing on task-oriented leaderboards. However, human feedback is notoriously vulnerable to manipulation, such as through sybil attacks.

AI's potential usefulness comes with vulnerabilities, and Alpha has adopted mitigation techniques that combine incentives, user reputation, and automated AI oversight to promote the collection of high-quality human feedback on interactions with Agents. Specific incentive structures will include community input, following Web3 best practices and with input from security experts. User reputation will include signals measurable within the protocol, such as past engagement, activity, and usage patterns. AI will assess the quality of human feedback, comparing feedback across users and evaluating authenticity.

At the protocol level, Human Evaluators are a class of Evaluator Agents that hold feedback provided by humans in order to produce quality signals. They may still use AI for internal processes, such as feedback summarizing and score tallying. Like AI Evaluators, Human Evaluator execution is carried out by the Alpha compute layer.

3.3 Custom Evaluators

Recognizing the need for focused, use case-specific quality signals, Alpha's Evaluators are designed to be extensible, supporting customization for narrower use cases. This feature is crucial for enabling users to tailor the protocol to their specific needs and for fostering innovation in evaluation methodologies.

For example, a user creating a Swarm to serve as a market intelligence assistant could develop a custom Evaluator by defining specific evaluation criteria, creating a set of representative tasks, and providing expected performance metrics for these tasks. This custom Evaluator can then be registered in the protocol to codify the user's specific requirements. We note that custom Evaluators – like any registered Agent – can be access-controlled at registration.

Users can create dynamic Swarms and optimize them with these custom Evaluators, enabling the definition of tailored evaluation criteria and leveraging Alpha's native Optimizers to identify or build Swarms suited to specific needs. This capability is particularly valuable in specialized domains or for users with unique requirements that may not be fully addressed by standard Evaluators.

3.4 Proof of Contribution

The work performed by both AI and Human Evaluators culminates in a Proof of Contribution – a cryptographically verifiable certificate that ensures the authenticity and integrity of the evaluation process. This proof comprises an encrypted, off-chain record of the Evaluator's execution (including inputs, logs, and outputs) and an on-chain hash of this record, creating a tamper-resistant evaluation record.

The Proof of Contribution serves a dual purpose. First, it provides users and Agents with verifiable information that contributes to an Agent's reputation. Second, and by extension, it helps to safeguard the optimization process against potential manipulation. While the protocol does not mandate a universal mechanism for end-to-end evaluation integrity due to the diverse nature of evaluation sources, it is designed to iteratively incorporate emerging best practices in evaluation integrity as Web3 technologies evolve.

4 Optimization in Alpha

Building upon the Evaluator framework discussed in the previous section, optimization is a fundamental mechanism that uses a combination of Agent reputation and stake to serve as quality signals for Swarm formation. This section explores the role of optimization in Alpha, introduces Bayesian optimization as a sample-efficient approach, and outlines future directions for generalizing Swarm membership determination.

4.1 The Role of Optimization in Alpha

In Alpha, optimization is the process of determining the optimal composition of a Swarm for a given use case, based on the results of Evaluators, subject to user-defined constraints, and influenced by Agent staking and delegation. Optimizers, specialized Agents in the protocol, improve Swarms by identifying optimal Agent compositions, particularly when a user has created an initial custom Swarm and wants to explore more effective configurations.

The objective function used in optimization is defined by a designated Evaluator, which may be a standard protocol Evaluator or a custom one created by the user. This flexibility allows for optimization across a diverse range of metrics, including but not limited to subjective quality, efficiency, cost-effectiveness, fairness, and accuracy. The use of custom Evaluators enables users to tailor the optimization process to their specific needs and preferences.

Before an Optimizer can run, it must identify (or be supplied with) a set of eligible Membership Candidates. In general, Agents in a Collective are eligible for substitution by any public Agent from any Collective that implements the same Behavior interfaces. The exact process an Optimizer uses to further determine the set of Membership Candidates is specific to each Optimizer. An Optimizer strategy may be to select the top eligible Agents from published leaderboards and taking direction from a user.

Figure 2 illustrates this process, showing how new versions of Agents can become Membership Candidates when they implement the same Behaviors and are registered to the same custom Evaluators as their predecessors. The figure also demonstrates how a Bayesian Optimizer can use a custom Evaluator that looks at analytics and accuracy to optimize the Swarm's membership.

Swarm optimization is essential for several reasons:

1. For users, it ensures that their experience using the protocol's Swarms is continually and automatically improving.

2. For developers, it provides a fair and transparent mechanism to gain visibility and drive utilization in the protocol.

3. Overall, it creates a competitive environment where Agents must add value to gain exposure to users.

4.2 Bayesian Optimization for Swarm Membership

To provide a sample-efficient reference Optimizer for the protocol, we propose the Bayesian Optimization by Tournament of Substitutions (BOTS) algorithm. Sample efficiency is crucial in this context due to the scaling costs of evaluating Swarm configurations. BOTS optimizes Swarm composition by building a predictive model based on past Agent evaluations and user stakes. This approach allows BOTS to estimate the performance of new configurations without exhaustive testing, significantly speeding up the process of finding high-performing Swarm compositions and doing so efficiently.

Let M be the set of all Membership Candidates, C the current Swarm, and $E : 2^M \to \mathbb{R}$ the Evaluator function, where 2^M denotes the power set of M (i.e., the set of all possible subsets of M). We employ a Gaussian Process (GP) as a surrogate model for the overall objective function, which may additionally consider constraints such as resource allocations. The algorithm uses an adjusted Expected Improvement (EI) acquisition function that incorporates both user stakes and stake diversity:

$$EI'(C^*) = EI(C^*) \cdot (1 + \log(1 + S_{C^*})) \cdot (1 + \alpha D_{C^*}) \tag{1}$$

where $EI(C^*) = \mathbb{E}[\max(E(C^*) - E(C^+), 0)]$ is the standard Expected Improvement, C^* represents a candidate Swarm configuration, C^+ is the current best-known Swarm configuration, $E(C^*)$ is the Evaluator's score for candidate Swarm C^*, S_{C^*} is the total stake on configuration C^*, $D_{C^*} \in [0, 1]$ is a measure of stake diversity (e.g., based on the distribution of stakes among different users), and α is a scaling factor. The logarithmic term for stakes and linear term for diversity balance the influence of large stakes while promoting configurations with broader community support. The Expected Improvement $EI(C^*)$ represents the expected value by which the new candidate Swarm configuration C^* will improve upon the current best observed value C^+, taking into account the uncertainty in the GP model.

The BOTS algorithm iteratively considers potential Swarm configurations, including both additions and removals of Agents, subject to predefined constraints (e.g., resource or size limits). It selects configurations based on EI', evaluates them using the Evaluator function, and updates the GP model with new information – ensuring that Swarm performance estimations are continually improving. This process balances exploration of new configurations with exploitation of known high-performing ones. The algorithm maintains the best-performing Swarm throughout the process and terminates when no further improvements are found or a maximum number of iterations is reached.

This approach, building upon established Bayesian optimization techniques [1, 7], efficiently navigates the configuration space while incorporating both the magnitude and diversity of community stakes. The continuous updating of the GP model allows BOTS to adapt to Alpha's dynamic ecosystem, where new Agents may be introduced and existing ones may evolve. By adapting these techniques to the unique challenges of optimizing AI Agent Swarms in a decentralized ecosystem, BOTS provides a robust framework for Swarm optimization that is particularly well-suited to Alpha's decentralized nature. A full description of the algorithm and its implementation will be available in a follow-up to this paper.

4.3 Generalizing Collective Membership

While the current implementation of Alpha optimizes fixed sets of Agents for Swarms, our long-term vision includes optimizing membership functions. This generalization will allow for more dynamic and context-aware Swarm formation while still maintaining the crucial concept of Proof of Collaboration.

A membership function would determine Agent inclusion based on task characteristics, historical Agent performance on similar tasks, current Agent availability, and user preferences. The process of shaping this function would itself generate a Proof of Collaboration, encapsulating the methodology used to determine task similarity metrics, the selection process for high-performing Agents on related tasks, and the optimization process for the membership function parameters.

This approach will enable Swarms to adapt their composition in real time, responding to changing needs and contexts. For example, a membership function could dynamically include Agents based on their past performance handling similar tasks or adjust the Swarm's composition to meet specific task requirements.

By supporting both direct set optimization and membership function optimization, Alpha aims to support a wide range of Aggregator behaviors, from simple routing to complex, context-aware Swarm formation. The Proof of Collaboration in this context would provide a verifiable record of the function's determination and supporting evidence.

This forward-looking approach will allow the protocol to evolve with the changing landscape of AI capabilities and user needs, enabling more effective and efficient Collective formation as the technology progresses, all while maintaining transparency and accountability through the Proof of Collaboration mechanism.

4.4 Proof of Collaboration and User Staking

When optimization results are encrypted and anchored on-chain, this constitutes a **Proof of Collaboration**. This proof serves as a verifiable record of a Swarm's membership determination process, ensuring its authenticity and integrity.

Proof of Collaboration encompasses:

- The optimized Swarm membership or membership function C^*
- The performance metric $E(C^*)$ associated with this membership
- A cryptographic hash of the optimization records and results
- An aggregated measure of user stakes for the constituent Agents

User staking plays a crucial role in the optimization process, particularly in the Bayesian approach. The integration of stake information into the adjusted acquisition function (EI') allows the optimization algorithm to consider both objective performance metrics and community sentiment when determining optimal Collective compositions. This dual input approach, combining technical performance and community assessment, is a core tenet in the development and deployment of trustworthy AI systems.

It is important to note that while user staking influences the optimization process, it does not solely determine the final outcomes, which are ultimately driven by an objective function to maximize the Evaluator score. This distinction ensures that Alpha promotes a meritocracy of Agents.

The Proof of Collaboration with staking represents a novel mechanism for validating and incentivizing effective agent collaboration in multi-agent systems. It aligns the interests of individual Agent developers, Swarm optimizers, and the broader user community, contributing to the overall health and efficiency of an AI ecosystem.

5 Alpha Protocol

Alpha is designed to unlock the potential of composability and interoperability for Swarms of AI Agents. This section provides insight into the protocol's architecture, core functionalities, and integration mechanisms within the broader Web3 ecosystem, with a focus on how Evaluators and Optimizers are integrated into the system.

5.1 Protocol Architecture

Alpha's architecture is rooted in the Ethereum ecosystem, leveraging smart contracts to implement core functionalities. The protocol employs a hybrid on-chain/off-chain model to optimize for scalability, cost-efficiency, and security, while ensuring interoperability and composability across diverse AI agents and frameworks. On-chain components, implemented via smart contracts, handle Agent registration utilizing non-fungible tokens (NFTs), token operations, and the anchoring of cryptographic proofs such as Proofs of Contribution and Proofs of Collaboration. These on-chain elements ensure transparency and immutability of critical protocol artifacts.

Complementing the on-chain layer, off-chain decentralized storage solutions such as Filecoin, 0g, Arweave, or Space and Time are employed for data-intensive components such as Agent Profiles, execution logs, and detailed Evaluator results. This approach maintains data integrity through cryptographic linking to on-chain anchors. The protocol's API layer facilitates seamless interaction between the blockchain infrastructure and off-chain Agents, enabling complex operations such as Agent invocation, configuration, discovery, registration, evaluation, and optimization.

5.2 Agent Development and Deployment

Alpha's design philosophy prioritizes flexibility and extensibility in Agent development. The protocol supports integration with a diverse array of AI models, compute environments, and Agent-building frameworks such as LangChain, AutoGen, ARC, Eliza, and CrewAI. Alpha also provides value-add Agent-building tools and reference implementations created by its core maintainers. This approach enables developers to leverage a wide array of cutting-edge AI technologies and tailor their Agents to specific use cases and performance requirements.

By abstracting the underlying infrastructure complexities, Alpha enables developers to use their preferred tooling for Agent-building while providing them with the, until now, missing interoperability and composability layers and an open marketplace. The protocol's extensible nature also allows for the creation and integration of custom Evaluators and Optimizers, enabling developers to define and optimize for specific performance criteria relevant to their use cases.

5.3 Verifiable Compute

As the Web3 technology stack evolves, Alpha is committed to progressively adopting verifiable compute within the protocol. Rather than relying on a single scalability solution, Alpha will leverage emerging layer-2 and multi-chain technologies to bring more protocol logic on-chain, enhancing transparency and verifiability while maintaining flexibility. End-to-end verification of all Agent computations remains challenging—especially for complex AI models—but Alpha will continue to monitor advances in cryptographic techniques such as zero-knowledge proofs and fully homomorphic encryption. In the interim, economic security models based on Byzantine Fault Tolerance (BFT) [3] and stake-weighted consensus provide strong incentives for correct and high-quality inferences while disincentivizing malicious behavior. As research matures and computational costs decrease, these cryptographic primitives will be integrated into the protocol to further strengthen the verifiability of off-chain computations.

5.4 Ecosystem Integration

Alpha is designed with interoperability as a core principle, aiming to integrate seamlessly with the broader AI and Web3 ecosystems. The protocol supports a wide range of AI models, data providers, and infrastructure providers, allowing users to select optimal tools for their specific use cases. A key focus of Alpha's integration roadmap is decentralized finance: Agents can interact with DeFi protocols, manage assets within AlphaVaults, and coordinate on-chain capital flows on behalf of users. Integration with Decentralized Physical Infrastructure Networks (DePINs) enables flexible deployment and management of AI Agents across diverse computational resources.

Furthermore, Alpha's architecture facilitates future interoperability with other blockchain networks and Web3 protocols. This multi-chain design enables seamless data and value transfer across ecosystems and positions Alpha as a nexus for AI-driven DeFi applications. The extensible nature of Evaluators and Optimizers in Alpha also allows for integration with external evaluation and optimization systems, further enhancing the protocol's adaptability and relevance in the evolving AI and crypto landscape.

6 Alpha Studio and Alpha Hub

Alpha Studio and Alpha Hub are two applications built upon the Alpha protocol to provide intuitive interfaces and rich functionality for users and developers. These applications make it easy for Agent consumers and application developers to interact with Agents, compose custom Swarms, and discover new Agents and services within the Alpha ecosystem. Both applications also facilitate Agent developers to register and monetize their Agents within the Alpha ecosystem. Alpha Studio and Alpha Hub leverage the protocol's modular architecture, allowing users to easily mix and match different Agents and components—including vaults and attribution layers—to create custom AI solutions.

6.1 Alpha Studio

Alpha Studio functions as a user interface application that enables users to interact with Agents through conversational interfaces and asynchronous Channels. The application supports session management for Agent interactions, allowing users to engage in long-lived conversations with various Swarms. A key feature of Alpha Studio is the Workspace interface, which lets users explore complex output objects, including visualizations, code snippets, data representations, and routing/planning internals. Future updates will incorporate publish/subscribe messaging within Channels, enabling Agents and users to exchange updates and notifications without requiring synchronous chat interactions.

The platform incorporates dedicated visualizations that offer insights into Agent operations during request processing. This feature enhances transparency and facilitates a deeper understanding of Agent behavior, decision-making processes, and financial performance metrics associated with AlphaVaults and DeFi strategies.

Alpha Studio's architecture allows for customization of the Agent discovery process, tailoring the user experience to specific requirements. Components such as session management, session creation, and the workspace are modular and can be configured out of the main view when a user prefers a simple chat or dialogue interface.

6.2 Alpha Hub

Alpha Hub serves as a marketplace and discovery platform for Agents and associated services within the Alpha ecosystem. It provides a user interface for browsing, searching, and interacting with Agents, as well as tools for developers to register and manage their Agents.

Key features of Alpha Hub include:

- Advanced search and filtering capabilities based on Agent capabilities, performance metrics, quality signals, and metadata
- Detailed Agent Profile information, including functionality specifications, pricing models, and historical performance data
- Integration of user reviews and ratings to facilitate quality assessment and reliability evaluation of Agents
- Seamless Agent discovery and interaction mechanisms for Alpha Studio
- Developer toolsets for Agent registration, updates, and performance monitoring

Alpha Hub offers convenient methods for composing new Agents and Swarms from existing components using templates, SDKs, and, in the future, graph-based agent-building tools. As the platform evolves it will support increasingly sophisticated customization and configuration options while maintaining an emphasis on developer ergonomics rather than no-code abstractions.

6.3 Developer Tools and SDK Integration

To facilitate the development and deployment of custom AI solutions, Alpha provides an extensive suite of developer tools. A production-ready Python SDK has been released, and JavaScript libraries are available for web integration. A Rust SDK is in active development to support high-performance use cases. These SDKs expose simple APIs for invoking Agents, managing sessions, and interfacing with on-chain vaults, enabling developers to embed agentic functionality into decentralized applications with minimal overhead.

Alpha also integrates seamlessly with established agent frameworks such as LangChain, ARC, Eliza, and CrewAI, allowing developers to leverage familiar tooling while gaining access to the Alpha protocol's marketplace and evaluation infrastructure. An increasing number of Agents are built atop the Modular Composable Protocol (MCP), which exposes a rich set of tools—including search, retrieval, finance, and contract execution—that can be invoked programmatically through the SDKs. These tools streamline the creation of sophisticated Swarms, connecting AI reasoning to on-chain actions and enabling rapid integration of new primitives as they are added to the protocol.

6.4 Composability and Extensibility

The modular nature of Alpha's architecture underpins the composability and extensibility of both Alpha Studio and Alpha Hub. By design, the protocol enables developers and users to easily compose, extend, and integrate various components within the ecosystem. This flexibility is achieved through simple, extensible Agent interfaces and protocol APIs that foster innovation and creativity, enhancing both interoperability between diverse AI systems and composability of complex AI solutions.

Developers can seamlessly create and integrate new Agents, Aggregators, Evaluators, or Optimizers into the protocol, enabling rapid innovation and adaptation to emerging AI technologies and use cases. This modular approach allows developers to build on the work of others, creating unique AI solutions that can be easily shared and monetized within the Alpha ecosystem.

The composability of the system is exemplified by the ability to enhance existing Swarms with new, specialized Agents. For instance, a developer could create a new 'Infographic Generator' Agent by combining existing Agents specializing in data pre-processing, data visualization, and data summarization. This new Agent could then be seamlessly integrated into existing Swarms, such as a 'Data Analyst' Swarm, enhancing its capabilities and potentially creating new revenue streams within the ecosystem.

This composability enables the rapid development and deployment of novel AI solutions tailored to the specific needs and interests of the community. As more specialized Agents are created and shared within the Alpha ecosystem, the possibilities for creating powerful, customized Swarms continue to expand, driving innovation in the burgeoning field of agentic AI.

7 THQ – Tokenomics in the Alpha Protocol

At the heart of the Alpha protocol is its native token THQ, which powers the emerging agentic economy. THQ has a fixed supply of one billion tokens and is designed to ensure long-term alignment among Agents, users, and developers. Rather than functioning as a generic payment currency or governance token, THQ serves as the value and coordination layer for an economy of intelligent, on-chain Agents. It underpins security through staking and slashing, aligns incentives through emissions and discounts, and fuels usage of the protocol through products such as AlphaSwarm—an AI-driven automation platform for high-value DeFi actions. Holders of THQ gain access to the Alpha protocol, can earn rewards by staking and locking their tokens, and can delegate their stake to Agents to power the ecosystem.

7.1 Staking, Locking, and Delegation

The Alpha token design introduces three core mechanisms for unlocking utility: staking, locking, and delegation. These mechanisms transform the base token THQ into two derivative forms—staked THQ (sTHQ) and locked THQ (αTHQ)—that confer rights, rewards, and responsibilities.

Staking ($\mathrm{THQ} \to s\mathrm{THQ}$). Users and Agents may stake their THQ to mint sTHQ, which locks economic value and secures the network. Stakers earn protocol emissions in THQ proportionally to their stake, and may also receive partner token distributions via AlphaSwarm. Staked THQ provides the baseline security layer for the protocol and underwrites Agentic activity; it can be withdrawn freely after a minimum staking period.

Locking ($s\mathrm{THQ} \to \alpha\mathrm{THQ}$). Holders may lock their sTHQ for a specified duration (between 1 and 24 months) within the **AlphaLocker** to mint αTHQ, a non-transferable representation of time-weighted power. Locked tokens earn enhanced emissions funded primarily by protocol fee buybacks (via Agentic activity in products such as AlphaSwarm) and augmented by partner token incentives. Locking provides strong alignment between long-term protocol participants and the network's growth. Upon expiry of the lock period, users regain their original sTHQ.

Delegation and Discounts. Holders of αTHQ can delegate portions of their balance to specific AI Agents. Delegating stake increases an Agent's ranking in discovery and optimization processes, improves its execution capacity, and entitles the delegator to discounted protocol fees when interacting with that Agent. Delegated αTHQ functions as slash-eligible collateral: if an Agent misbehaves or underperforms, the delegated stake (and the underlying sTHQ) may be slashed, tightening supply and reinforcing security. Delegators share in the protocol and agent-specific rewards generated by the Agent.

Rewards and Participation. Participants in the Alpha economy—stakers, lockers, delegators, liquidity contributors to AlphaVaults, and Agent operators—earn rewards from protocol fees, emissions, and partner incentives. Liquidity contributors deposit assets into on-chain vaults managed by Agents and receive fees and yields from automated strategies. Agent developers earn protocol emissions and fee shares for providing valuable services. Across all roles, THQ provides access and discounts for using the protocol.

7.2 Integration with Protocol Operations

Stakes and delegations are deeply integrated into the Alpha protocol's core processes. In optimization algorithms such as BOTS, staked and delegated balances influence the acquisition function and, consequently, Swarm membership decisions. Proofs of Collaboration embed token stake and delegation data, providing verifiable signals of community confidence in specific Agent configurations. Delegated αTHQ directly affects an Agent's discovery ranking and execution capacity, while also serving as slash-eligible collateral in the event of misbehavior. Tokens may also be used to incentivize participation in Human Evaluator tasks, ensuring a steady supply of high-quality human feedback. Through these mechanisms, THQ becomes the economic and reputational fabric that aligns incentives and drives continuous improvement within the Alpha network.

8 Governance and Safety

Alpha's approach to AI safety begins with robust AI governance rooted in high ethical standards, continuous monitoring and evaluation, and testing and validation, leveraging both AI- and human-based evaluation to ensure trustworthy AI. All Agents benefit from Alpha's built-in safety and quality mechanisms, including spam detection and prevention and a robust set of built-in Evaluators.

We believe that AI Evaluators will play an increasingly critical role in safety over time. AI Evaluators enable the creation of metrics that detect and discourage non-compliant or malicious behavior. These automated systems can continuously monitor Agent behavior, providing real-time safety assessments. Human Evaluators can complement this approach by offering nuanced, context-aware safety evaluations that can capture subtle ethical considerations or potential misuse scenarios that may be challenging for AI systems to identify.

Recognizing that current AI systems still exhibit unpredictable behaviour [5], Alpha encourages ongoing research and collaboration with AI safety experts to proactively identify and mitigate risks. Future developments may include the creation of dedicated Safety Evaluators, combining both AI and human insights, to monitor and assess the safety of Agents and Swarms more comprehensively.

The protocol includes mechanisms for removing Agents that violate community standards or pose security risks. In an open community, the determination of removal criteria is an important governance consideration that will include community and expert engagement. Overall, Alpha aims to create a secure and trustworthy environment for AI Agent interactions and collaborations, setting a standard for responsible AI development in decentralized ecosystems. To this end, Alpha is committed to an iterative governance model, continuously evolving its approach through collaborative research, community input, and expert engagement to address the dynamic challenges of decentralized agentic AI.

8.1 Security and Privacy

Alpha implements a comprehensive approach to security and privacy within its ecosystem. The protocol incorporates a range of mechanisms to mitigate risks and promote responsible AI development, including:

1. Staking and slashing mechanisms to discourage malicious behavior
2. Robust security audits and bug bounties to identify and address vulnerabilities

3. Social dispute resolution and arbitration processes
4. Collaboration with leading AI safety researchers and organizations
5. Integration of AI and Human Evaluators to continuously assess and improve the safety of Agents and Swarms

To enable effective governance, Alpha's architecture leverages a hybrid model, combining on-chain and off-chain components to optimize for scalability, cost-efficiency, and data protection. Access control is managed through Public Key Infrastructure (PKI) and granular permissions, ensuring that sensitive information remains accessible only to authorized entities.

Data storage utilizes both on-chain storage for immutability and transparency, and decentralized off-chain solutions for scalability. On-chain storage primarily maintains links to decentralized data and critical, immutable information, while encrypted Agent activities and user interactions are stored off-chain. Execution logs are encrypted and stored within Profiles, accessible only to designated key holders, enabling necessary auditing while maintaining privacy.

As the protocol evolves, Alpha remains committed to adopting emerging best practices in cryptographic security and privacy-preserving technologies to ensure the highest standards of data protection and user trust. The protocol may evolve to support multi-signature key scenarios for enhanced security in specific use cases, such as dispute resolution or safety auditing.

9 Roadmap and Timeline

The development of the Alpha protocol and its tokenomics is structured in clearly defined phases to ensure sustainable growth and resilience. These phases outline the rollout of staking, locking, delegation, and extended functionality across multiple chains and DeFi verticals. Feedback from all stakeholders—including early users, developers, and community members—will guide the refinement of each phase.

Phase 1: Staking Core (Mainnet). The initial phase launches the staking core on mainnet, enabling holders to stake THQ and receive $s\mathrm{THQ}$. Baseline emissions begin, establishing the security layer of the network. Agents can begin executing on-chain strategies via AlphaSwarm on Base, the first supported chain, with additional EVM chains to follow.

Phase 2: Lock-Up & Security. The second phase introduces the **AlphaLocker** and mints $\alpha\mathrm{THQ}$. Users can lock their $s\mathrm{THQ}$ for durations between 1 and 24 months to earn enhanced emissions funded by protocol fee buybacks and partner incentives. This phase also activates slashing mechanisms to compensate for protocol failures or malicious behavior, subject to approval by a super-majority of $\alpha\mathrm{THQ}$ holders. Additional EVM chains will be brought online as part of this phase.

Phase 3: Delegation & Agent Modules. The third phase enables delegation of $\alpha\mathrm{THQ}$ to specific Agents, unlocking discounts, boosting agent capacity, and activating slash-eligible collateral. New agent modules—including capital-managing modules for AlphaVaults and cross-chain executors—are deployed, and on-chain fee splitting is introduced to improve composability. Multi-chain support continues to expand, enabling Agents to operate seamlessly across Base and other EVM networks.

Future Phases. Subsequent phases will focus on further decentralization, establishing protocol insurance reserves, introducing innovative reward mechanisms, and expanding the utility of THQ as the agentic economy matures. Additional integrations with DeFi protocols, Layer-2 networks, and real-world assets are envisioned, along with continued research into verifiable compute and cryptographic assurances.

9.1 Community-Driven Safety

The success of Alpha's safety efforts ultimately depends on the active engagement and participation of the community. User feedback and participation play a crucial role in maintaining the safety and security of the Alpha ecosystem. Users can report potential vulnerabilities, flag malicious or unethical behavior, and provide suggestions for improving the platform's safety measures.

By fostering a culture of transparency, collaboration, and shared responsibility, community members contribute to the development of safety standards, best practices, and risk-mitigation strategies. This stewardship may involve the community proposing and voting on guidelines for creating safe Agents, including requirements such as extensive testing, adherence to ethical principles, and the incorporation of fail-safe mechanisms. These guidelines can then be integrated into both AI and Human Evaluators, ensuring that safety standards are consistently applied and monitored across the Alpha ecosystem. Additionally, the community may establish a dedicated Safety Council, composed of experts in AI safety, ethics, and governance, to guide the implementation of these standards and provide ongoing support to developers and users.

10 Conclusion

Alpha is a novel protocol for creating and managing AI Agents and Swarms that addresses critical gaps in today's AI and DeFi ecosystems. By providing robust solutions for interoperability, composability, capital management via on-chain vaults, and a multi-chain infrastructure, Alpha unlocks the potential of collaborative agentic intelligence and on-chain automation.

What sets Alpha apart is its flexibility and extensibility coupled with a dynamic marketplace and an incentive-driven token model. The protocol is designed to evolve with technological progress and user interests, supporting long-lived, asynchronous Agents that operate across multiple chains and integrate seamlessly with DeFi primitives. Through tools such as AlphaSwarm and AlphaVaults, developers and users can build powerful agentic applications that not only reason but also act on chain.

Core to Alpha's vision is its commitment to community-driven innovation. The ecosystem invites contributions from developers, researchers, and AI enthusiasts, offering SDKs and integration with popular frameworks such as LangChain, ARC, Eliza, and CrewAI. Participants can build new Agents and Swarms, contribute feedback, stake, lock, and delegate THQ to support agents, and deposit assets into vaults. This alignment of incentives fosters a vibrant agentic economy where value flows freely and performance is rewarded.

We invite you to be part of this exciting journey in shaping the future of decentralized, agentic intelligence and finance. For more detailed information about Alpha, its development roadmap, and engagement opportunities, please visit theoriq.ai, join our Discord community, and follow updates on X @TheoriqAI. Together, we can unlock the full potential of AI and DeFi responsibly.

References

[1] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[2] C. Burns, P. Izmailov, J. H. Kirchner, B. Baker, L. Gao, L. Aschenbrenner, Y. Chen, A. Ecoffet, M. Joglekar, J. Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

[3] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[4] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C. Qian, C.-M. Chan, Y. Qin, Y. Lu, R. Xie, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2023.

[5] J. Clark and D. Amodei. Faulty reward functions in the wild, 2016. Accessed: 2024-06-18.

[6] I. Gabriel et al. The ethics of advanced ai assistants, 2024.

[7] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pages 937–945, 2014.

[8] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[9] T. Masterman, S. Besen, M. Sawtell, and A. Chao. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*, 2024.

[10] X. Zhou, H. Zhu, L. Mathur, R. Zhang, H. Yu, Z. Qi, L.-P. Morency, Y. Bisk, D. Fried, G. Neubig, and M. Sap. Sotopia: Interactive evaluation for social intelligence in language agents, 2024.