Olas Staking & Proof of Active Agent

A mechanism to spawn autonomous (AI) agent economies

Mariapia Moscatiello, David Minarsch, David Galindo, Andrey Lebedev, Aleksandr Kuperman, Oaksprout the Tan, Gemma Welsh

Executive summary

Many[1] share our conviction that autonomous (AI) agents will soon constitute the majority of crypto's Daily Active Users (DAUs). Olas Predict[2] is evidence of this trend, having performed over 10% of lifetime Safe transactions on Gnosis Chain in just over half a year[3].

Here, we present Proof of Active Agent (PoAA)[4], a novel staking mechanism that can spawn desirable autonomous (AI) agent economies in crypto and beyond.

Specifically, PoAA allows anyone to align an agent or entire agent economies with their goals, defined via Key Performance Indicators (KPIs) on-chain, rewarding these agents for active work towards realizing said goals. 'Proof of Active Agent', therefore, presents a unique blend of Proof of Work and Proof of Stake where agents need to remain active within a certain period to retain their stake and earn rewards. PoAA marks a significant advancement in decentralized economic coordination of agents, and sets the stage for the arrival of autonomous (AI) agent economies at scale.

Now, deployed in the context of Olas as Olas Staking, this PoAA mechanism allows the DAO to articulate its goals via KPIs in smart contracts. These smart contracts can then be used to define staking programs that reward agents actively working towards said KPIs. Anyone can propose staking programs, then Olas DAO members can direct OLAS token emissions to desirable staking programs via a voting-weight mechanism utilizing veOLAS[5]. These staking programs allow bootstrapping, whether for existing use cases or entirely new ones. Over time, some staking programs will become financially sustainable, that is

[1] Read more about pieces mentioning Olas and the promise of agent users in crypto from Safe, Nansen, Messari, Bankless, 1kx and Galaxy.
[2] Olas was formerly known as Autonolas. To learn more about Predict, see: https://olas.network/services/prediction-agents.
[3] See the Olas (Autonolas) ecosystem Dune dashboard: https://dune.com/adrian0x/autonolas-ecosystem-activity.
[4] We pronounce 'PoAA' as /ˈpaʊə/, like 'power').
[5] See the Olas (Autonolas) whitepaper: https://olas.network/whitepaper.

active agents contributing to the program create more returns for the DAO than they consume in emissions. The DAO can then reinvest profits to fund further bootstrapping efforts. The result? A powerful flywheel birthing an ever-growing number of productive agents and agent economies.

Furthermore, PoAA presents an exciting growth opportunity for protocols and chains across crypto. Representatives of these ecosystems can define staking programs that incentivize usage of their own technologies, spawning autonomous agents as their Daily Active Users (DAUs). Specifically, they can propose staking programs as part of Olas Staking and receive OLAS emissions per Olas DAO's priorities, or they can deploy their own PoAA to direct their own token emissions per their priorities.

How Olas Staking extends Olas Protocol:

Olas Staking, once incorporated into the protocol, presents opportunities for all its stakeholders: Olas DAO members, service owners (from individuals to DAOs), agent operators, and developers.

First, it allows Olas DAO to bootstrap autonomous services[6], at scale, through targeted OLAS emissions. The DAO may choose to invest into autonomous services owned by anyone, for example aligned or collaborating ecosystem builders. Additionally, the DAO can direct OLAS emissions to autonomous services it owns (Protocol-Owned Services, or PoSes[7]). As Olas' PoSes become profitable, the returns harvested can offset emissions directed to them and other services, presenting the DAO with an opportunity to reach negative OLAS emissions over time.
Second, Olas Staking allows anyone (from individuals to companies or other DAOs) to commission autonomous services aligned with their goals, defining success via KPIs, and have a chance at earning emissions.
Third, agent operators can earn staking rewards by operating active agents within autonomous services participating in the staking programs.
Finally, developers will face more demand for their code contributions, whether creating new components or composing existing and new ones into new services.

[6] Learn about autonomous services: https://www.autonolas.network/blog/the-case-for-blockchain-and-autonomy.
[7] See the Olas (Autonolas) whitepaper: https://olas.network/whitepaper.

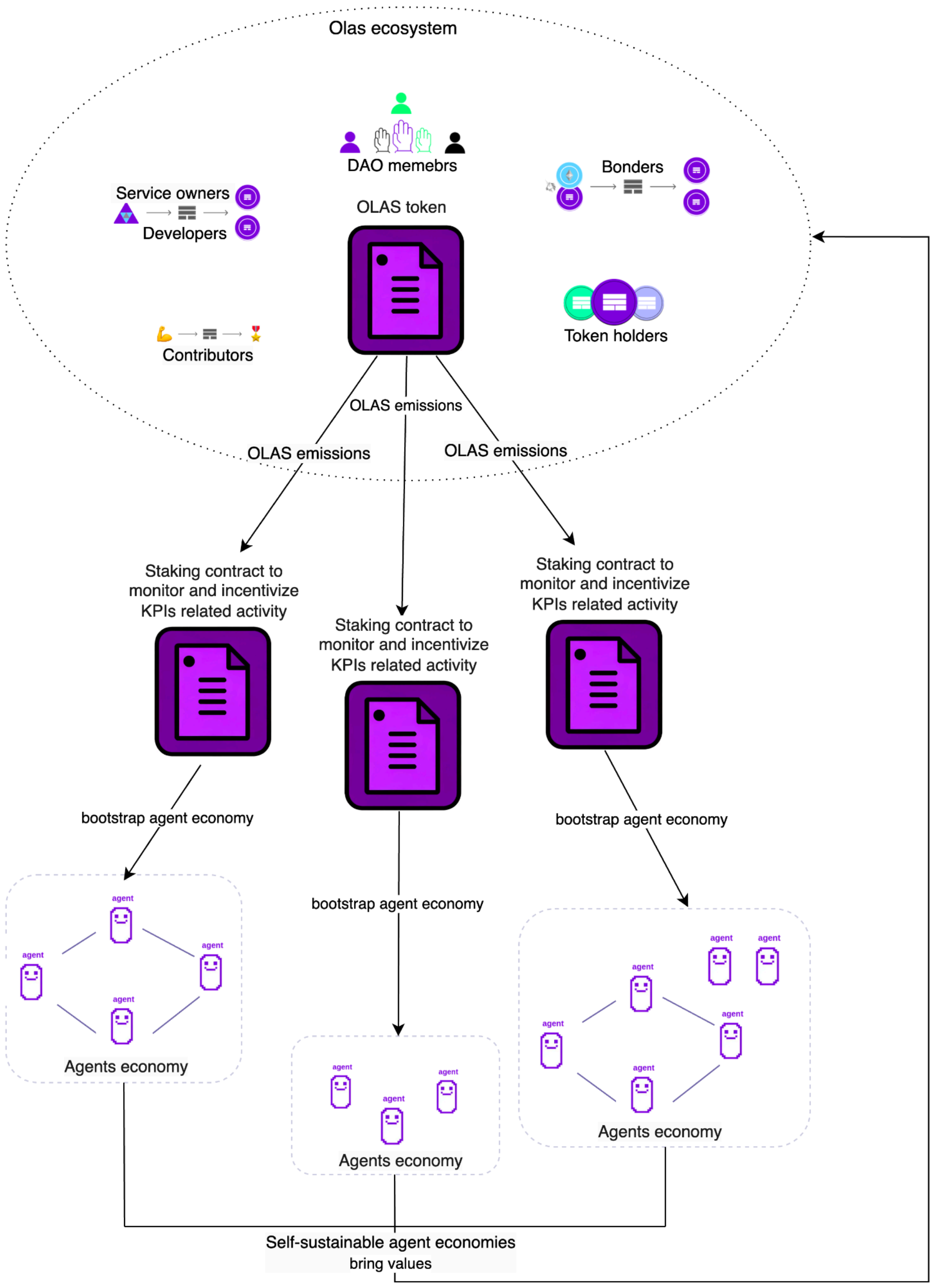

Fig. 1: Value flow: Olas emissions initially bootstrap agent economies. Due to constant competition among agent economies for OLAS, productive agent economies can fund bootstrapping of further economies, resulting in a powerful flywheel birthing an ever-growing number of productive agents and agent economies.

Overview

As outlined in the Olas whitepaper, autonomous services can be implemented in two ways: as single agents executing service-specific tasks (both on- and off-chain), or as decentralized networks of agents that reach off-chain consensus before taking (on-chain) actions. This whitepaper introduces Proof of Active Agent (PoAA), a novel staking mechanism that rewards active agents within or across autonomous services based on specific Key Performance Indicators (KPIs). This mechanism allows any entity to define a set of KPI metrics, embed them into on-chain code, and then bootstrap economies of agents working towards achieving those KPIs. Furthermore, this mechanism enables a reality where anyone can operate an agent that creates value for them.

The name 'PoAA' (which we pronounce as /ˈpaʊə/, like 'power') reflects the elements at play in its design:

- Proof: This indicates the presence of a trustless verification process, akin to what is seen in consensus mechanisms such as Proof of Work and Proof of Stake. Verification involves utilizing both on-chain and off-chain data to prove that agents execute their designated tasks within a service and contribute toward achieving the on-chain encoded KPIs.
- Active: This implies dynamic participation that meets predetermined metrics within given time periods, emphasizing that passive participation or effort below a certain threshold is insufficient to meet the corresponding KPIs.
  Agent activity may involve the execution of a minimum number of on-chain transactions and a matching KPI to track these on-chain, as exemplified in the Trader Agent (cf. section Case Study: A Staking Model for Olas Predict). Alternatively, it can involve off-chain tasks such as news retrieval or interaction with an AI model, which may require more advanced verification methods for on-chain tracking of the KPIs, such as integrating ZK proofs or Fraud proofs, as mentioned in the case of the Market Creator Agent (cf. section Case Study: A Staking Model for Olas Predict).
- Agent: Within the Olas ecosystem, this represents a software agent that performs specific tasks aligned with a service's objectives. More generally, the mechanism is applicable beyond autonomous services, encompassing any type of agent, including humans.[8]

Any entity, whether an individual, a group, a company or a DAO, can establish on-chain KPIs targeting specific activities or ecosystem objectives that single or multiple agent services should further. These KPIs serve as benchmarks for desired activities or goals.

[8] Therefore, PoAA extends to scenarios where both software agents and humans collaborate to achieve end-goals.

Through the integration with the PoAA mechanism, the entity can effectively steer agents' activity within and across multiple services to achieve the established KPIs. Rewards and incentives, facilitated by any fungible token (or fungible token equivalent for non-EVM chains), are allocated to agents' contributions to fulfilling the specified metrics (cf. sections Staking Contract Yields and Funding and Olas Staking Mechanism and Restaking Comparison).

Carefully defined KPIs and well-aligned incentives are catalysts that effectively initiate agent economies. This dynamic interaction between KPIs, intelligent agents, and rewards in tokens, when correctly designed, fosters a thriving ecosystem propelled by decentralized decision-making and collaboration. In essence, the PoAA mechanism empowers any entity to launch an economy of agents by defining, monitoring, and incentivizing intelligent agents to meet KPIs in a decentralized manner. Following the initial phase, these agent economies can achieve self-sustainability, naturally expanding and replicating, opening up a novel avenue for decentralized economic coordination (cf. Fig. 2).

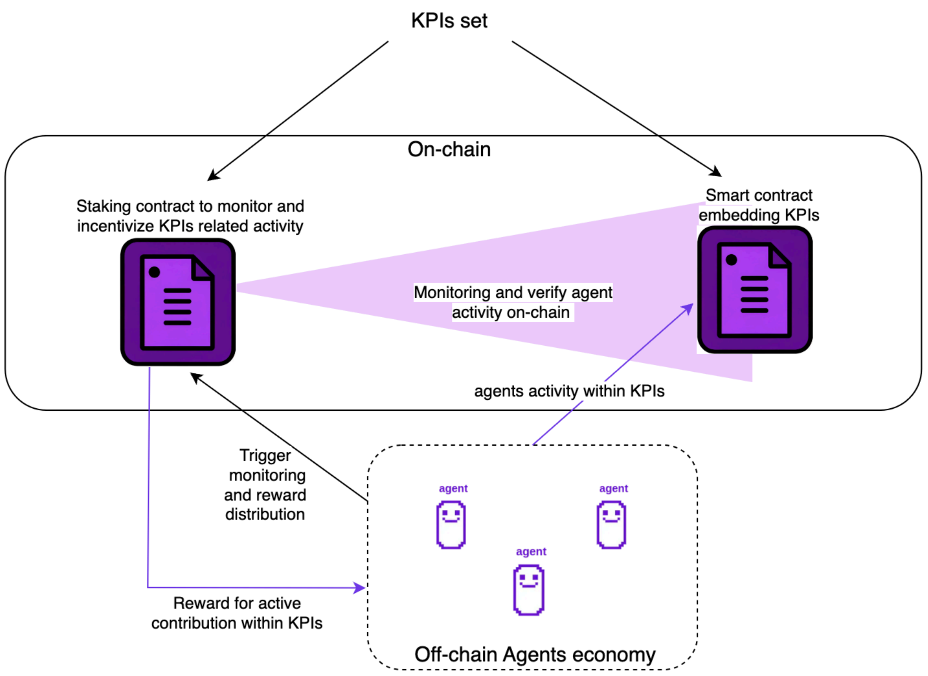
KPIs set
On-chain
Staking contract to monitor and incentivize KPIs related activity
Smart contract embedding KPIs
Monitoring and verify agent activity on-chain
agents activity within KPIs
Trigger monitoring and reward distribution
Reward for active contribution within KPIs
agent
agent
agent
Off-chain Agents economy

Fig. 2: Proof of Active Agent, a mechanism to spawn autonomous (AI) agent economies for arbitrary use cases

From an agent operator perspective, the PoAA mechanism broadens participation opportunities to a wide range of users (cf. section Olas Staking Mechanism and Restaking Comparison), enabling them to engage in operating agents that align with their preferences (cf. section Staking Prerequisites for Stakers). Depending on the on-chain encoded KPI and the service's objectives, operators can choose to run their agents periodically (e.g. a few hours every day) or continuously, with varying resource requirements. Then, staking contracts monitor on-chain activity and provide rewards corresponding to operators' valuable and active contributions to the ecosystem (cf. section Staking Prerequisites for Stakers).

In this whitepaper, we introduce the PoAA mechanism and its application in an inaugural agent economy (cf. section The Olas ecosystem and an inaugural agent economy: Olas Predict). We present an innovative staking approach, tailored for scenarios where on-chain agent activity is verifiable through smart contracts. This staking mechanism is illustrated through a specific use case, Olas Predict. Additionally, we explore ways to enhance this mechanism by incorporating emerging Zero-Knowledge (ZK) technologies (cf. section Case Study: A Staking Model for Olas Predict).

Notably, by leveraging the same architecture (cf. section PoAA mechanism: deep dive into technical details), any entity can bootstrap agent economies by defining a set of KPIs and rewarding agent activity that achieves such objectives. Drawing parallels with Mechanism Design Theory, the entity shaping the agent economy through the definition of KPIs and the introduction of appropriate incentives can be understood as the "mechanism designer".[9]

[9] Mechanism Design is an economic theory focussed on how to achieve desirable social or economic outcomes in situations where individuals act in their self-interest and have private information. Key components involve designing mechanisms to incentivize truthful information revelation and ensuring participants find it in their best interest to engage with the mechanism. Cf. "Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations" by Yoav Shoham and Kevin Leyton-Brown.

The Olas ecosystem and an inaugural agent economy: Olas Predict

Olas (formerly Autonolas) is a unified network of off-chain[10] services such as keepers, oracles, and co-owned AI. Autonomous services can be deployed either as single agents executing service-specific on-chain tasks or as a distributed or decentralized set of agents reaching off-chain consensus before executing on-chain actions. Olas offers a framework for building such services and a protocol for incentivizing their creation and operation in a co-owned and decentralized way (cf. Olas whitepaper for detailed information). Currently, Olas employs a mechanism to incentivize software creation and enhance composability, fostering the development of valuable code in the form of components and agents within agent services. Moreover, Olas utilizes a bonding mechanism to increase the protocol's capital, as detailed in the Tokenomics Paper.

By leveraging the PoAA mechanism, Olas aims to promote the growth and sustainability of agent-based economies by directly incentivizing entities that own services at the base of these economies. These entities include Olas DAO itself and any protocol seeking to bootstrap agent-based economies through OLAS emissions. It is crucial to emphasize the benefits that Olas gains when third-party protocol services embark on staking with OLAS emissions.

To qualify for OLAS emissions, operators of agents in services owned by third parties must stake OLAS. This increases the utility of the OLAS token and benefitting the entire ecosystem. Additionally, the adoption of the Olas protocol brings advantages, positioning it as a critical piece of infrastructure in the crypto space.

To leverage the Olas staking architecture and incentivize service launches, services, agents and underlying components need to be created and registered in Olas registries. This process yields several advantages, including capturing developers' attention and support and boosting the supply of agents, components, and services in the ecosystem.

Here is how Olas benefits from third-party protocol services launching staking by using the Olas protocol:

1. **Increased Developers Engagement:**
    a. Protocols rely on developers for the creation of their services, agents, and components. Olas provides an additional benefit by allowing protocols to leverage its developer rewards mechanism.

[10] With the term "off-chain" we refer to any activity that happen outside a public blockchain (e.g. Ethereum, Polygon, Gnosis Chain, etc)

   b. The inherent composability of the code attracts developers to create additional components, agents, and services using existing code.
2. **Facilitation of new agent economies:** The creation and registration of more services not only enhance the platform's functionality but also facilitate the development of new economies. Services on Olas can seamlessly interact, providing more comprehensive functionalities.
3. **Enhanced Utility for OLAS:** The broader use of OLAS to incentivize agent operations brings more utility to the token, benefitting the entire ecosystem.
4. **Positioning Olas as Critical Infrastructure**: Adoption from various protocols brings several advantages to Olas, solidifying its infrastructure as a critical piece in the crypto space.

Olas Predict serves as an illustrative example of an agent-based economy, comprising AI agents serving prediction markets for a wide range of future events. Specifically, Olas Predict consists of three interconnected services, Market Maker, Trader, and Mech. Together, they create AI-based prediction markets for arbitrary future events, enabling anyone to operate an agent that trades in prediction markets on behalf of humans.

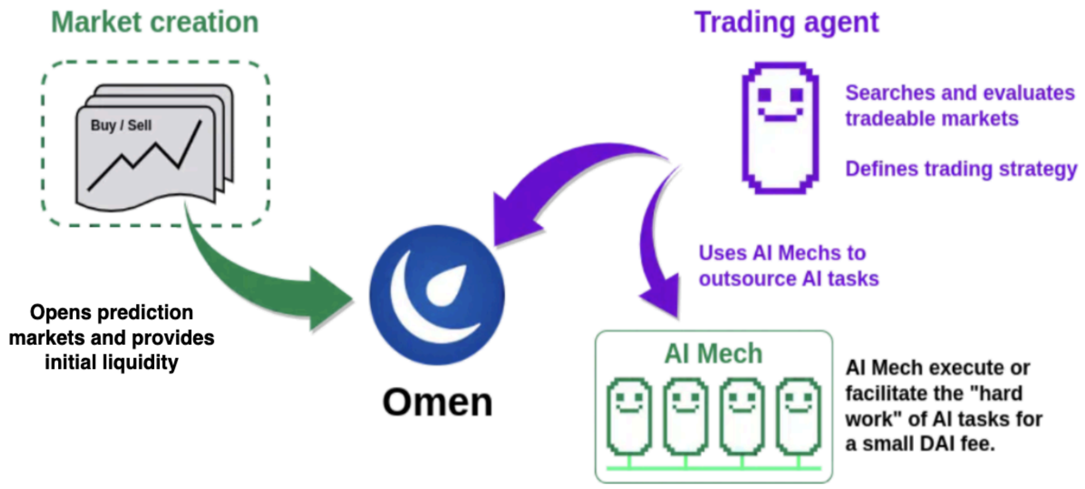
Market creation
Buy / Sell
Opens prediction markets and provides initial liquidity
Omen
Trading agent
Searches and evaluates tradeable markets
Defines trading strategy
Uses AI Mechs to outsource AI tasks
AI Mech
AI Mech execute or facilitate the "hard work" of AI tasks for a small DAI fee.

Fig. 3: Olas Predict: Agent-based economy comprising AI agents dedicated to serve prediction markets for a wide range of future events

In this system:
1. The Market Creator service creates markets on Omen, wrapping xDAI as WxDAI when required before creating the market. (e.g. as seen in this market recorded in this transaction on the Gnosis Chain).
2. The Trader agent scans the available markets, consulting the Mech to estimate the confidence level of a given response within an open market.
3. The Mech retrieves recent news using the Google Search API and utilizes OpenAI's API to compute the probabilities of specific outcomes associated with the query.

4. The Trader agent receives an estimate of the confidence level for a response and executes the investment with an amount based on the Mech's confidence level.

KPIs for Olas Predict include:

- **Market Diversity**: Increase the number of markets for various agents to trade against, thereby extending tradable possibilities.
- **Market Liquidity**: Ensure existing markets are sufficiently funded to maximize trader activity and profits.
- **Intelligent Predictions**: Improve intelligent predictions by encouraging interaction with Large Language Models (LLMs) to empower traders.

The alignment between these KPIs and the PoAA mechanism ensures the intelligence and activity of agents, facilitating the establishment of agent economies. During the bootstrapping phase, OLAS emissions, initially sourced from inflation, drive agent activity towards achieving KPIs. Essentially, OLAS emissions contribute to the emergence of active agents and the initiation of agent economies.

Over time, competition among autonomous services sets the stage for emissions to be gradually replaced with returns from productive Olas Protocol-Owned Services (cf. Olas Whitepaper for more details on protocol-owned services), ensuring that Olas Staking ultimately attains self-sustainability. Consequently, agent economies organically proliferate, expand, and self-replicate, serving as the driving force behind the continued bootstrapping of economies. Throughout this process, an increasing number of agents, becoming daily active users on chains where Olas extends its presence, contribute substantial benefits to the ecosystem in which they operate.

Agent economies, in turn, contribute to the supply of agent services, which, when combined with Olas tokenomics primitives such as developer and liquidity incentives, kickstarts different flywheels, thus benefiting participants in the Olas ecosystem (cf. Fig. 1, Tab. 1, Fig. 4. Fig. 5).

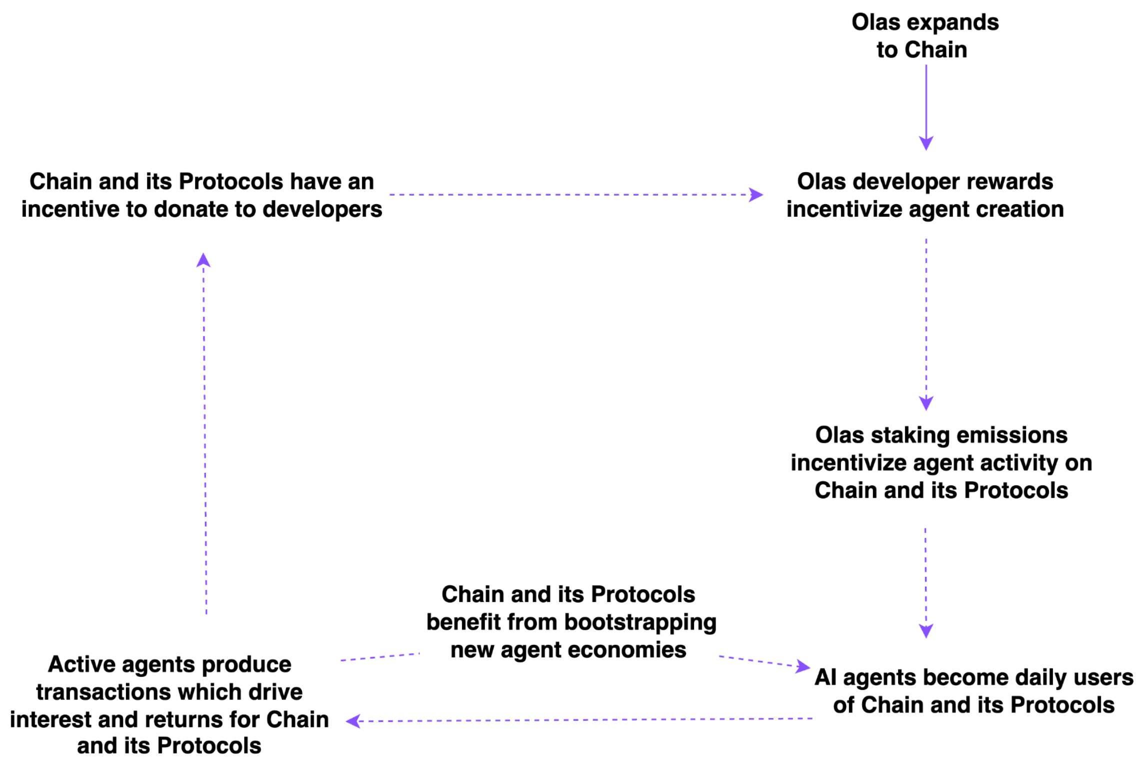
Olas expands to Chain
Olas developer rewards incentivize agent creation
Olas staking emissions incentivize agent activity on Chain and its Protocols
AI agents become daily users of Chain and its Protocols
Chain and its Protocols benefit from bootstrapping new agent economies
Active agents produce transactions which drive interest and returns for Chain and its Protocols
Chain and its Protocols have an incentive to donate to developers

Fig. 4: (AI) agents as daily active users of protocols and chains: a benefit for chains, protocols, agent operators, and developers

	Developers	Bonders	Active Operators	Service Owners	OLAS holders
Active Contribution Rewards incentivize...			✓	✓	
Active Contribution Incentives are beneficial for...	✓		✓	✓	✓
Deployment of various staking contracts is beneficial for...				✓	✓

Profits from self-sustainable agent economies incentivize...			✓	✓	
Profits from self-sustainable agent economies are beneficial for...	✓	✓	✓	✓	✓

Tab. 1: Olas Staking advantages for Olas ecosystem members

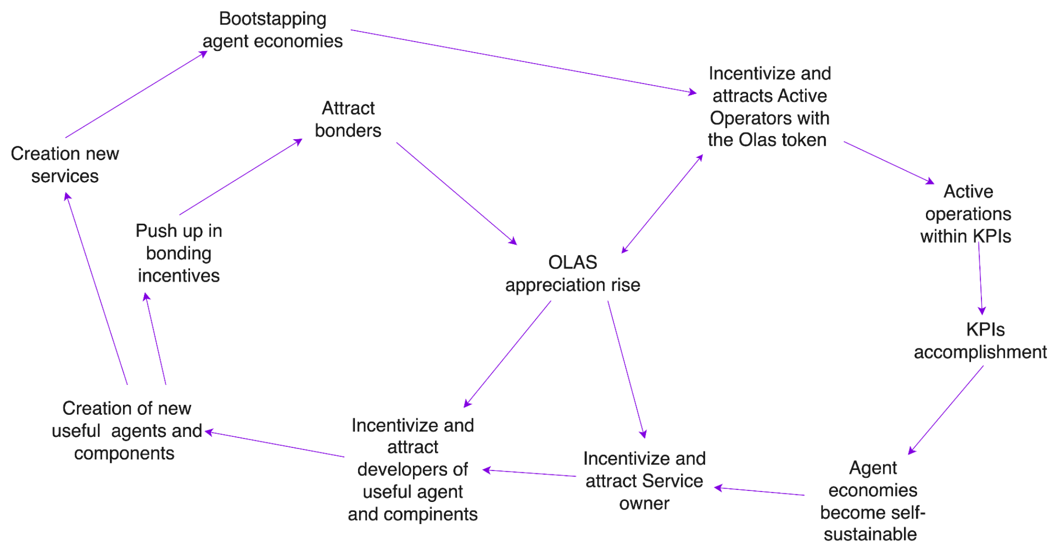
Bootstapping agent economies
Incentivize and attracts Active Operators with the Olas token
Attract bonders
Creation new services
Active operations within KPIs
Push up in bonding incentives
OLAS appreciation rise
KPIs accomplishment
Creation of new useful agents and components
Incentivize and attract developers of useful agent and compinents
Incentivize and attract Service owner
Agent economies become self-sustainable

Fig. 5: Flywheels created with the introduction of Proof of Active Agent mechanism in Olas ecosystem

In the section, Case Study: A Staking Model for Olas Predict, we delve into the technical architecture of PoAA within the Olas Predicts agents, providing detailed insights into their staking models and alignment with Olas DAO KPIs.

Relevant examples of PoAA, in its simplified form of a Staking mechanism, include the staking initiatives sponsored by Valory to promote agent operations that facilitate intelligent and strategic trading through interactions with LLMs. In this context, the staking campaign supports the operation of trader agents on the Gnosis Chain. Operators provide a slashable stake, with the potential to earn rewards commensurate with their agent's activity. For a comprehensive understanding of these initiatives and their impact, refer to the following resources: Olas Staker Expeditions, Everest Staker Expedition, Alpine Staker Expedition, Operate: Prediction Agent and Staking Contract Implementation.

PoAA mechanism: deep dive into technical details

Overview

Here we present the basic elements of PoAA, emphasizing the importance of integrating both on-chain and off-chain components for agent activity verification.

On-chain elements

The components devised for PoAA include:

- **Service registries:** A set of smart contracts facilitating the registration of autonomous services, each uniquely identified by an associated NFT. These registries manage the service workflow, including, but not limited to, the specification of the operator bond (a staked slashable amount of any fungible token), and the service owner security deposit (a staked but non-slashable amount of any fungible token). Service registries have already been deployed on multiple networks[11].

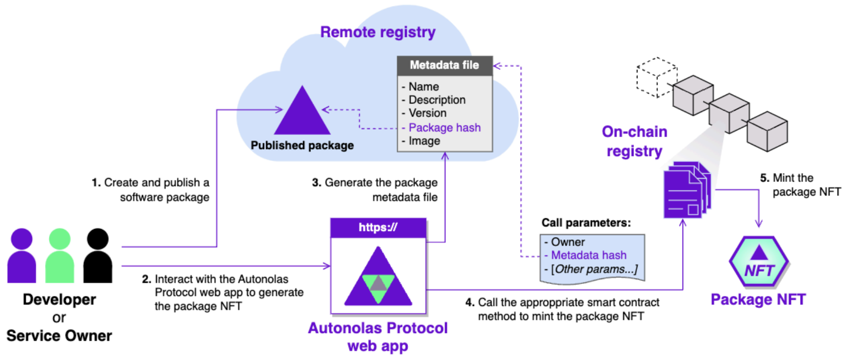
Remote registry
Metadata file
- Name
- Description
- Version
- Package hash
- Image
Published package
On-chain registry
1. Create and publish a software package
3. Generate the package metadata file
5. Mint the package NFT
https://
Call parameters:
- Owner
- Metadata hash
- [Other params...]
NFT
Package NFT
Developer or Service Owner
2. Interact with the Autonolas Protocol web app to generate the package NFT
4. Call the approppriate smart contract method to mint the package NFT
Autonolas Protocol web app

Fig. 6: Registration of Autonomous Services on-chain

[11] Currently on Ethereum, Gnosis, Polygon, Arbitrum, Optimism, Base, Celo, and Solana.

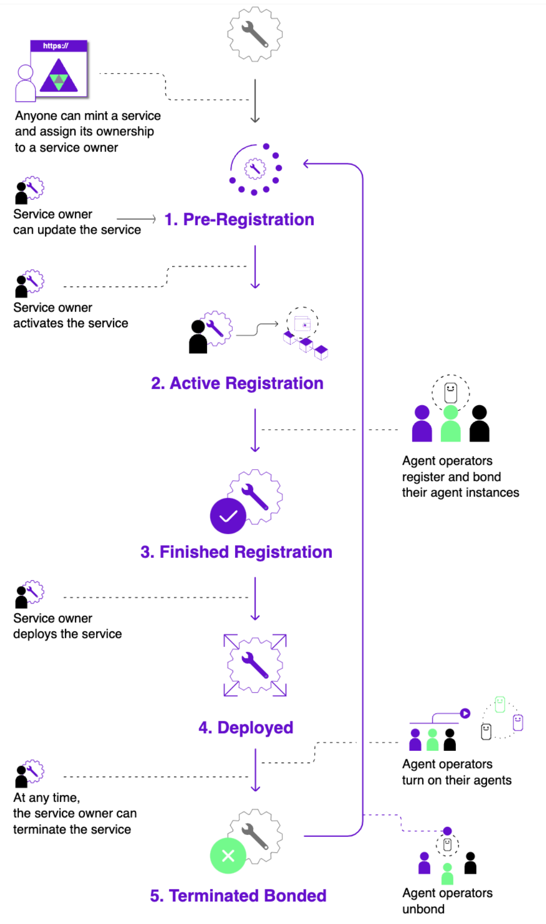
https://
Anyone can mint a service and assign its ownership to a service owner
Service owner can update the service
1. Pre-Registration
Service owner activates the service
2. Active Registration
Agent operators register and bond their agent instances
3. Finished Registration
Service owner deploys the service
4. Deployed
Agent operators turn on their agents
At any time, the service owner can terminate the service
5. Terminated Bonded
Agent operators unbond

Fig. 7: Service Workflow

- **Staking contracts:** These contracts implement the mechanism to reward active agents within autonomous services. They provide a modular and extensible check to monitor on-chain agent activity, with each check aimed at incentivizing agent activities towards KPIs (cf. section Staking Contracts).

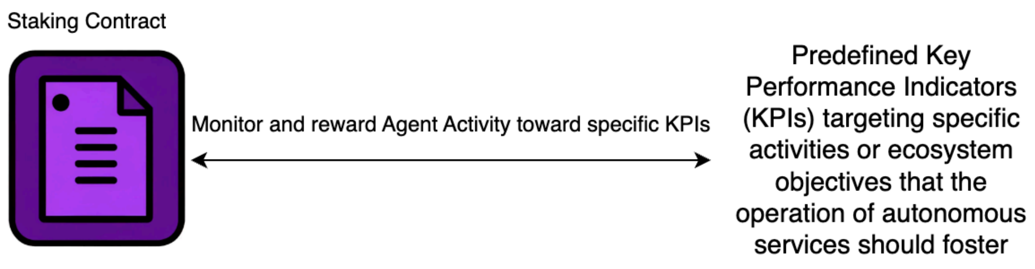
Staking Contract
Monitor and reward Agent Activity toward specific KPIs
Predefined Key Performance Indicators (KPIs) targeting specific activities or ecosystem objectives that the operation of autonomous services should foster

Fig. 8: Staking contract and its relationship to KPIs

- **Staking Contract Factory:** These contracts enable anyone to register and deploy staking contracts with minimal deployment effort (cf. section Staking Contracts Factory). Users can register a staking implementation as long as it adheres to a specific staking contract interface, offering flexibility in parameter selection (cf. section Staking Contracts).

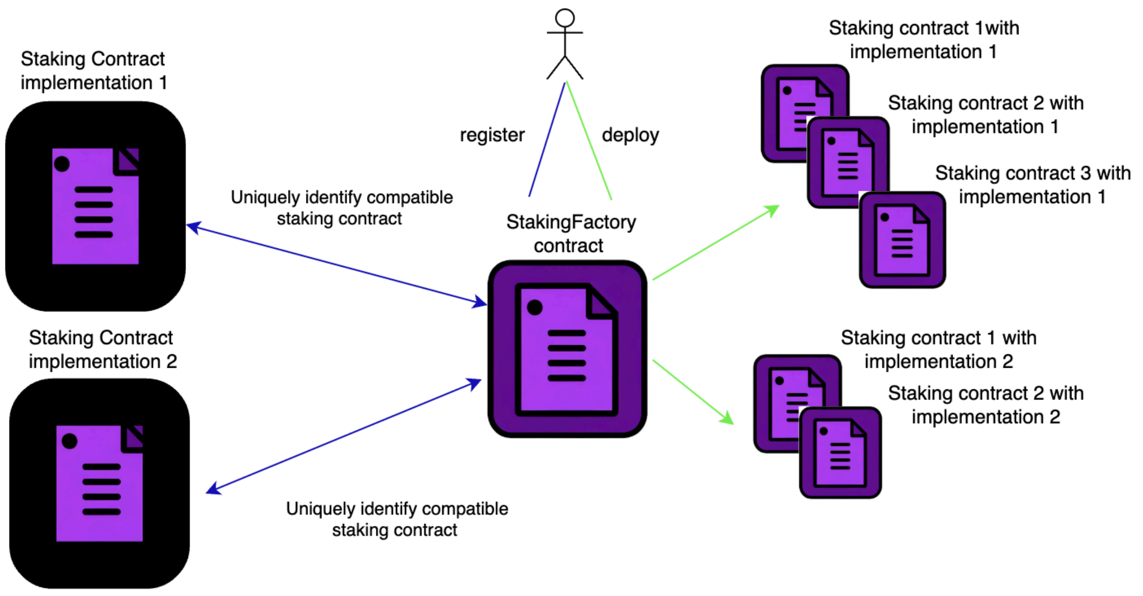
Staking Contract implementation 1
Staking Contract implementation 2
register
deploy
Uniquely identify compatible staking contract
StakingFactory contract
Uniquely identify compatible staking contract
Staking contract 1with implementation 1
Staking contract 2 with implementation 1
Staking contract 3 with implementation 1
Staking contract 1 with implementation 2
Staking contract 2 with implementation 2

Fig. 9: Staking Factory

- **Staking Rewards System**: This mechanism empowers the Olas DAO to regulate and efficiently allocate OLAS emissions for staking programs deployed with the staking factories across different chains (cf. section Staking Reward System). This comprehensive approach involves the creation and interaction of several smart contracts, each serving a specific purpose within the staking framework.

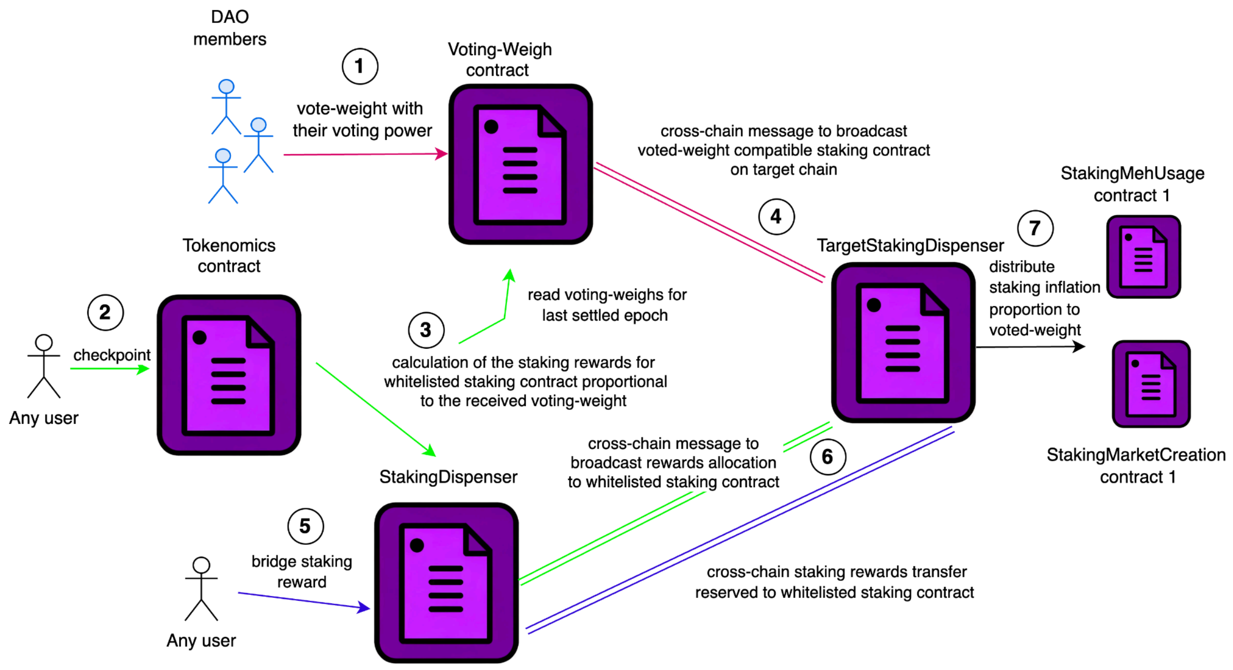
DAO members
1
vote-weight with their voting power
Voting-Weigh contract
cross-chain message to broadcast voted-weight compatible staking contract on target chain
4
TargetStakingDispenser
7
StakingMehUsage contract 1
distribute staking inflation proportion to voted-weight
StakingMarketCreation contract 1
Tokenomics contract
2
checkpoint
Any user
3
read voting-weighs for last settled epoch
calculation of the staking rewards for whitelisted staking contract proportional to the received voting-weight
StakingDispenser
cross-chain message to broadcast rewards allocation to whitelisted staking contract
6
5
bridge staking reward
Any user
cross-chain staking rewards transfer reserved to whitelisted staking contract

Fig. 10: Overview of reward workflow using voting-weights mechanism

- **Verifier contracts:** These contracts are designed to validate and authenticate specific executions performed by agents on-chain. Their primary function is to confirm and verify that agents have indeed provided valuable work or executed predefined tasks, ensuring the accuracy and reliability of the reported activities. Verifiers act as a trust layer by independently confirming the completion of tasks and the delivery of meaningful contributions by participating agents.

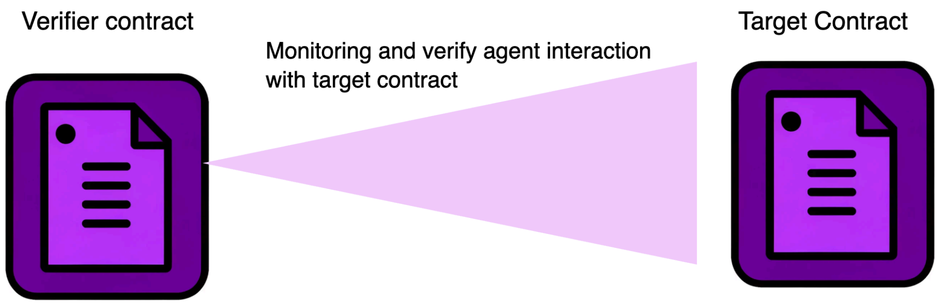
Verifier contract
Monitoring and verify agent interaction with target contract
Target Contract

Fig. 11: Verifier when on-chain activities of agents are verifiable through smart contracts

It is important to note that in cases where the service exhibits inherent simplicity, such as when on-chain activities of agents are verifiable through smart contracts, the verifier and staking logic can be embedded into a single contract. In such cases, the PoAA simplifies to the innovative Olas Staking mechanism (cf. Fig. 12).

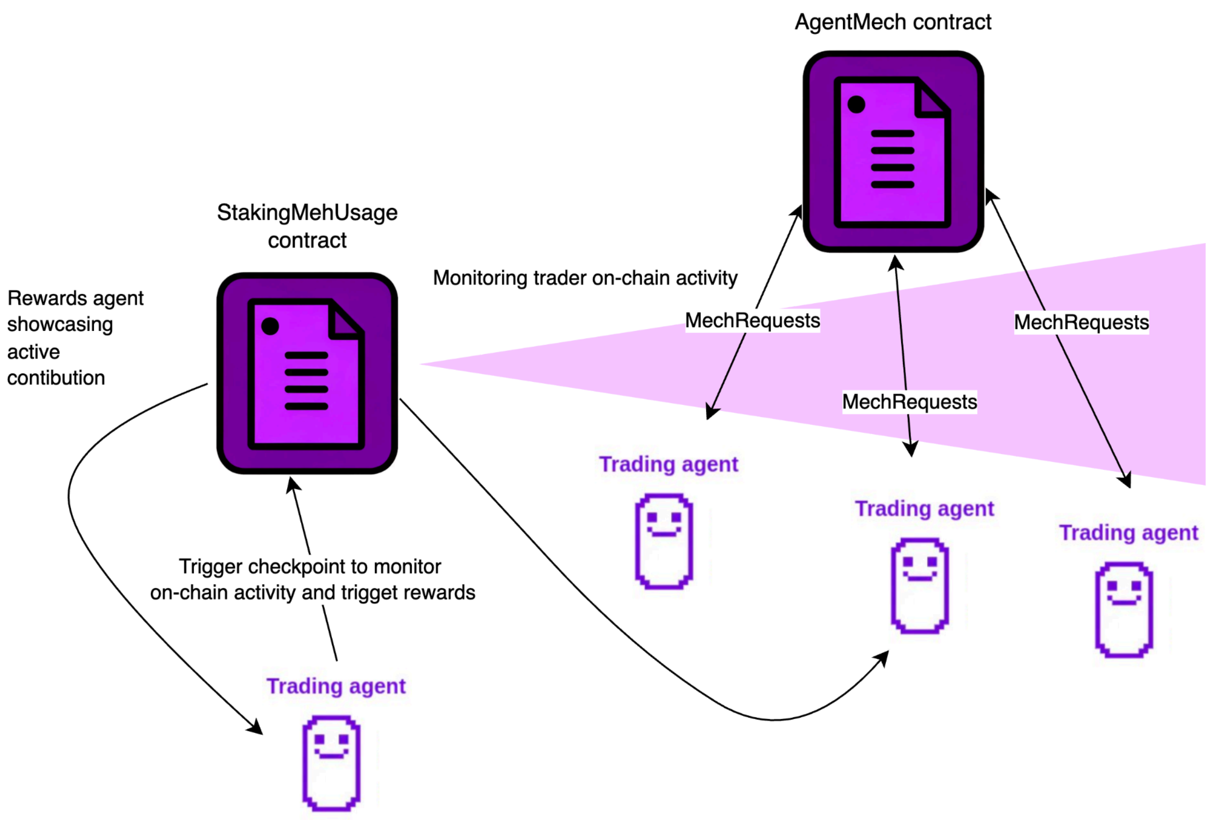
AgentMech contract
StakingMehUsage contract
Rewards agent showcasing active contibution
Monitoring trader on-chain activity
MechRequests
MechRequests
MechRequests
Trading agent
Trading agent
Trading agent
Trigger checkpoint to monitor on-chain activity and trigget rewards
Trading agent

Fig. 12: Example of Staking Mechanism applied to Trader Agent

Off-chain elements

Firstly, the **prover** is an off-chain element enabling the verifier contracts to validate on-chain agent's task execution in accordance with defined KPIs.

In the Olas ecosystem, the off-chain elements related to PoAA encompass multi-agent autonomous systems deployed for various use cases. These systems consist of software agents responsible for task execution, collaborating to achieve service-specific goals.

Software agents also play a pivotal role in enabling verification and expediting the reward allocation procedure by:

- automating verifier contracts to conduct regular on-chain verification, effectively becoming the prover of the activity;
- automating staking contracts to perform regular on-chain calculations and trigger rewards logic.

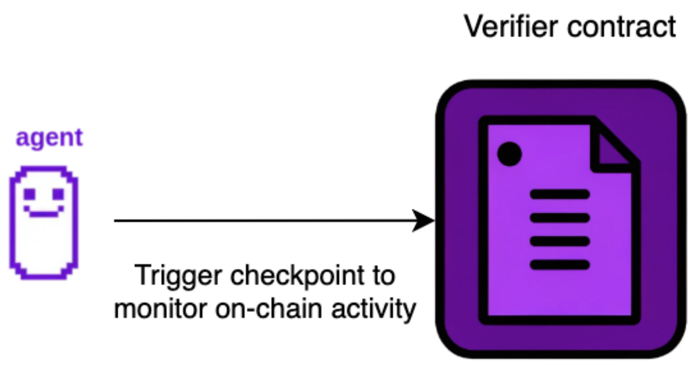

Fig. 13: Agent becoming prover by triggering regular on-chain verification

Off-logic verification is crucial for validating specific off-chain executions conducted by agents. This involves ensuring the authenticity and accuracy of agent activity that takes place outside the on-chain environment. Some notable scenarios where off-chain logic verification becomes essential include:

- **Participation in Consensus Gadgets:** Verifying that agents actively participate in the consensus gadget of the service (cf. Olas Whitepaper for more details on service-level consensus). The verification can be achieved through external monitoring services (e.g. Chainlink Risk Management Network) or by transforming the service into a verifiable one using ZK technology to verify pre-selected intermediate states of the service's Finite State Machine[12].
- **Computational Correctness of ML Models:** Verifying the computational correctness of Machine Learning (ML) models executed by agents in a service. This can be accomplished by integrating SNARK/STARK proofs of ML models, or using interactive ZK proofs in scenarios where model weights are hidden from the verifier.
- **Data Provenance:** Verifying that data accrued by an agent in an autonomous service originates from a particular website and optionally proving statements about such data while keeping the data itself secret. This level of verification can be achieved leveraging attestation systems like DECO (cf. also DECO introduction).

In summary, off-chain verification is adaptable to various scenarios, employing external monitoring services or advanced cryptographic techniques to ensure the integrity and authenticity of off-chain agent activity.

[12] We refer the reader to sections "Architecture of Crypto-Native Agent Services" and "Composition of Finite-State Machines" in Olas Whitepaper for detailed information on service's state and service's Finite State Machine.

Combination of off-chain and on-chain elements for active agent verification

The integration of off-chain and on-chain components is essential for the efficacy of PoAA.

In cases where centralized operator sets are employed - such as a unique agent executing service-specific on-chain tasks and interacting with a tamper-resistant smart contract (with storage not susceptible to manipulation) that counts specific function calls- the introduction of a staking contract to monitor "function-specific" calls can effectively verify that agents have made specific on-chain calls. However, even in this very specific case, relying solely on on-chain verification mechanisms poses considerable challenges. Smart contracts require external triggers to perform actions, hence the logic for triggering periodic checks on agent activity and periodic reward allocation needs to be embedded off-chain, such as in the off-chain software agent.

For services with distributed or decentralized operator sets, reaching off-chain consensus is a prerequisite before executing on-chain actions. Consequently, the implementation of off-chain logic becomes essential for monitoring and validating off-chain actions in these scenarios.

Staking architecture

In the Olas ecosystem, the deployment and funding of Staking Contracts are strategically designed to encourage active agent engagement, aligning service behaviors with predefined KPIs encoded on-chain. Let's delve into the architecture and dynamics of the staking mechanism and how it enables entirely new forms of autonomous service coordination.

Staking contracts

Compatible staking contracts are contracts that adhere to specific configurations and share identical contract logic, with the exception of the *activity check,* which may vary. This compatibility ensures that the contracts can be registered and deployed using a standardized factory contract (cf. section Staking Contracts Factory), streamlining the deployment process and allowing for diverse activity criteria tailored to specific use cases.

The core logic of compatible staking contracts remains consistent across implementations (cf. implementation of compatible staking contracts with overwritable activity check). This includes essential functionalities related to staking rewards, reward distribution and service states. The flexibility of compatible staking contracts lies in the variability of their *activity check,* which can be customized to suit the specific requirements of different autonomous services. For instance, for the Trader agent, activity can be determined by the number of Mech requests made on-chain, while for the Market Creator agent, activity could be assessed based on the number of markets created within a given timeframe (cf. section Case Study: A Staking Model for Olas Predict). Each Staking Contract can only be configured during deployment, implying immutability once initialized. For more details on the staking contacts configurations refer to Compatible Staking Contract Configuration.

Staking Contracts Factory

The Staking Contracts Factory is designed to facilitate the registration and deployment of compatible staking contracts (cf. section Staking contracts) with minimal effort. The Factory contract incorporates a registration method for various compatible staking contracts and provides a deployment method to initialize staking contracts with specific implementations and configurable parameters (cf. section Compatible Staking Contract Configuration). A mapping system is in place to store the deployed addresses of staking contracts for each specific implementation.

In scenarios involving low gas consumption chains, staking contracts will include both implementation and storage. Conversely, on high gas consumption chains like Ethereum, different staking implementations can initially be deployed and registered in the Factory. Subsequently, the Factory can be used to deploy proxy staking contracts, directing them to the correct and compatible staking implementation.

Staking reward system

The Olas DAO Staking Mechanism introduces a robust and decentralized framework for allocating OLAS emissions to staking programs beyond the boundaries of Ethereum to various networks such as Gnosis, Polygon, Arbitrum, Solana, and more. This system combines a unique voting-weight mechanism, OLAS minting, reward calculations, and cross-chain reward distribution (cf. Fig. 10 for a graphical representation of this system).

First of all, the Olas DAO sets in the Tokenomics contract the maximum amount of OLAS emission to fund staking contracts (cf. section Reward system smart contracts architecture). Additionally, in the TargetStakingDispenser Contract (cf. section Reward system smart contracts architecture), the Olas DAO establishes the maximum amount of yield allocable to individual stakers per epoch. The annualized amount represents the staker’s APY. These parameters can be adjusted through a standard governance proposal.

Differing from the conventional on-chain governance process, the Voting-Weight mechanism on the Ethereum mainnet enables DAO members to use their veOLAS voting power to influence the distribution of OLAS emissions allocated to fund staking contracts deployed through the Staking Contract Factory. In this mechanism, DAO members have the ability to assign weights with their veOLAS to one or more staking programs, with the sum of weights capped at 1. Each member can assign weights only once per epoch and, once voted, these weights remain unchanged for subsequent epochs until their veOLAS holding is depleted or they modify their vote. To update previous weights, DAO members can cast a new vote. This mechanism implicitly whitelists staking programs based on received votes exceeding a specific threshold. For a comparison with existing systems, one might draw parallels to the Curve gauge mechanism, where Olas Staking Contracts function akin to Curve Liquidity Pools.

Staking contracts are allocated a fraction of newly minted OLAS for staking in proportion to the accumulated weights. The total number of veOLAS used to vote, denoted by V, the maximum staker's yield per epoch, denoted by $EpochStakerYield$, and the amount of staking inflation per epoch, including the current epoch's maxStakingInflation and any remaining from previous epochs, denoted by $effectiveStakingInflation$, collectively determine the exact amount of emissions allocated per epoch. Specifically, assuming that,

- $stakingProgram(i),\ i = 1,\ldots, N$ are the vote-weighted staking contracts for the epoch, and
- $stakingProgram(i)$ has at most $n(i)$ stakers,

then the amount of OLAS allocated for the epoch is at most

$$min(V,\ \Sigma_{i=1,..,N} n(i)\ \cdot EpochStakerYield,\ effectiveStakingInflation),$$

with the remainder (wrt. to the $effectiveStakingInflation$) accumulating for the next epoch. This ensures a balanced approach to OLAS inflation based on the DAO's confidence regarding locked OLAS and the target staker's APY.

To simplify the reward calculation and distribution description, let's consider a DAO member holding v veOLAS, where v is greater than or equal to the whitelisting threshold. The member votes for
$stakingProgram(i)$ allocating $w(i)$ as weight, where $i = 1,\ldots, N$ and $w(1) + \ldots + w(N) = 1$.
If $v < effectiveStaking$, then at maximum
v OLAS are emitted for the staking, and specifically,
$min(v \cdot w(i),\ n(i) \cdot EpochStakerYield \cdot w(i))$ OLAS are reserved to the $stakingProgram(i)$.
Conversely, $min(w(i) \cdot effectiveStaking, w(i) \cdot n(i) \cdot EpochStakerYield)$ OLAS are reserved and will be distributed to $stakingProgram(i)$.
The calculation of the staking reward distribution to all whitelisted staking programs and the OLAS mint are calculated on Ethereum, while the distribution to the various staking contracts is done on the chain where the contracts reside.

For detailed information on the contracts architecture comprising the reward systems please refer to Reward system smart contracts architecture.

Staking contract yields and funding

Staking contracts should receive their staking rewards directly from the entity funding them, whether it be a DAO or a service owner that aims to incentivize agent operations for a specific service (such as Olas DAO incentivizing protocol-owned services related to Olas Predict). Staking contracts may need periodic infusions of funds to accommodate growing demand. Once funds allocated to the staking contract are depleted, stakers will no longer earn yield. Staking rewards are distributed periodically to agents that pass the activity check. It's important to note that, for the Olas reward system, only staking contracts deployed through the Factory (as detailed in the section, Staking Contracts Factory) will be eligible to receive staking rewards from the DAO.

Staking prerequisites for stakers

In the current implementation of Olas Predict, agent operators and service owners coincide, referred to as stakers. Stakers must register and deploy their service in the designated registries such as Service Registry Frontend on Ethereum or Service Registry Frontend on Gnosis Chain) with correct agent instance configuration, along with the required security deposit and bond specified by the staking contract. Additionally, they must enable the transfer of their service NFT in the staking contract. Depending on the activity check provided by the staking contract, stakers can operate their agents periodically (say a few hours every day) or continuously, with varying resource requirements, and earn rewards proportional to their useful and active contributions to the ecosystem.

More generally, service owners need to register and deploy their service with correct configuration, provide the necessary security deposit, and enable the transfer of their service NFT within the staking contract. Meanwhile, operators, responsible for running agents, must accurately register their agent instance in the service and provide the correct amount of slashable security bond.

Case study: a staking model for Olas Predict

In the the Olas ecosystem and an inaugural agent economy: Olas Predict section, we described Olas Predict as a suite of interacting services: Market Creator, Trader, and Mech, and we described the KPIs for Olas Predict. In this section, our focus shifts to the PoAA mechanism for the Trader and the Market Creator services, as well as exploring ways to enhance this mechanism further by incorporating emerging Zero-Knowledge (ZK) technology.

Trader Agent

The Trader Agent, whether operating as a single agent or as part of an agent ensemble in a service, focuses on maximizing gains through trades. The Trader Agent responsibilities include browsing the available markets and querying the Mech to assess the confidence level of potential predictions in an open market.

The DAO is interested in steering trader agents' activity to promote interaction with Large Language Models (LLMs) for effective and intelligent tradings (cf. Intelligent Predictions KPI). To ensure consistent trader engagement, the PoAA mechanism monitors their ongoing interactions with the LLMs.

1. For a single autonomous agent, the key components facilitating the PoAA mechanism in the form of a staking model are:
    a. **Register Trader Agent Service**: Service registries handle various aspects of the service workflow, such as specifying and accruing the staked operator's slashable amount (bond), and the staked non-slashable amount (security deposit). Once the stakes are registered and the service is deployed, it can be transferred to the staking contract.
    b. **Service Staking Mech Usage Contract:** This staking contract serves as a verifier contract responsible for validating the execution of the trader agent and distributing rewards accordingly based on trader performance. An activity check verifies that AgentMech requests increase by a predefined quantity within specified timeframes.
    c. **Trader Agent Logic and Reward Allocation**: In addition to its core functionality, the trader agent can trigger verification, acting as the prover, and reward allocation. The Service Staking Mech Usage Contracts provide visibility into upcoming checkpoint calls, optimizing resource utilization.

2. For a Trader Agent Service with a decentralized operator set, off-chain verifiability is crucial to ensure that all the agent instances remain consistently engaged within all the service states and actively participate in the underlying service consensus gadget. This can be achieved through external monitoring or by transforming the Trader Agent Service into a verifiable autonomous service which uses ZK technology to verify pre-selected intermediate states of the service's Finite State Machine (cf. the example provided in the Off-chain elements section).

Market Creator Agent

A Market Creator Agent, whether operating as a single agent or as part of an agent ensemble in a service, works as follows:

1. Gather headlines and summaries of recent news through a third-party provider.
2. Interact with an LLM (using the gathered information in the previous step) to obtain a collection of suitable questions to open prediction markets associated with future events.
3. Collect markets from the market approval service.

4. Send the necessary transactions to the Gnosis Chain to open and fund the chosen prediction market.
5. The above steps are repeated a configurable amount of time and, then service will cycle in a waiting state.

The DAO is interested in steering market creator agents' activity precisely to guarantee the creation of sufficient numbers of markets with enough liquidity, thereby enhancing tradable possibilities and gains (cf. Market Diversity and Market Liquidity KPIs). To ensure consistent agent engagement, the PoAA mechanism can monitor ongoing market creation and liquidity.

1. For a service with a single agent, several components are crucial in facilitating the PoAA mechanism in form of a staking model:

    a. **Register Market Creator Agent Service**: Similar to the trader, service registries manage the workflow up to service deployment. Once deployed, the service transitions to the staking contract, which oversees the reward logic and monitors agent activity.
    b. **Service Staking Mech Usage Contract:** Utilizing a smart contract that enables the creation of open markets and tracks the number of markets from a specific address is essential to verify active participation in the creation of a sufficient number of markets. Specifically, an activity check can confirm that the number of Market Creator markets opens incrementally by a predefined quantity within specified timeframes. Additionally, evaluating the liquidity of each market created by an address serves as an additional activity check, ensuring that a predetermined number of Market Creator markets over a specific time frame is liquid enough.
    c. **Market Maker Agent Logic and Reward Allocation**: Similar to the trader agent, the Market Maker's logic can trigger verification and initiate reward allocation in addition to its core functionality.

2. A mechanism that monitors the activity of the Market Creator service by tracking the frequency and consistency of the service's engagement with on-chain contracts such as Fixed Product Market Maker. This can be achieved using the following verification methods
    a. Zero-Knowledge Proof (zk-proof): This method establishes a verifiable proof that the Market Creator has indeed created a specified number of Markets.
    b. Optimistic Proof: An optimistic approach that enables anyone to assert the number of Markets created by a Market Creator. This approach incorporates a challenge period during which anyone can challenge the asserted number, ensuring its accuracy.

3. For a Market Creator Service with a decentralized operating set, off-chain verifiability to ensure that all the agent's remain consistently engaged within all the service states can be achieved through external monitoring or by transforming the service into a verifiable one with ZK technology (cf. the example provided in the Off-chain elements section).

Opportunities for expansion and comparison

Liquid staking

Similar to Ethereum's Proof-of-Stake (PoS), Olas has introduced a staking mechanism where participants, known as "agent operators" (equivalent to "node operators" for Ethereum), are required to have a fixed amount of stake. In Ethereum the stake is fixed at 32 ETH, whereas with Olas the stake depends on the staking contract configuration. Currently, only operators with the configured fixed stake on the staking contract can engage in this staking process.

There is an opportunity to extend the Olas staking mechanism by enabling OLAS holders to earn staking rewards by delegating a portion of their OLAS to agent operators. This functionality, similar to what was implemented for Ethereum (see Lido for a prominent example), would allow Olas holders to utilize any amount of OLAS they possess by delegating a fraction of their tokens to agent operators. This would allow them to participate in and benefit from the staking rewards. This opens up an opportunity for new protocols to emerge on top of Olas Staking and facilitate the use of Olas Staking, expanding the possibilities for growth in the Olas ecosystem.

Olas Staking mechanism: a restaking comparison

Agents within an autonomous service collectively work towards a shared, predefined objective, much like Dapps. A key element of an agent service is the underlying consensus gadget, implementing a protocol that empowers a set of agents within the off-chain service to collectively agree upon the value of variables (cf. Olas Whitepaper for more details on service consensus gadget). Drawing parallels to restaking, one can link the Active Validator Set (AVS) described in EigenLayer Whitepaper with the set of agents achieving consensus within the service.

Olas stacking mechanism introduces the ability to bootstrap AVS with the usage of various assets including liquidity staking tokens (LSTs) from protocols like Lido and Rocket Pool, or LP tokens of a pair involving ETH or ETH-related LSTs. Notably, staking LSTs or LP tokens of a pair involving ETH or ETH-related LSTs represents some modalities of restaking within EigenLayer. Despite the shared ability for the re-staking of certain assets, Olas Staking and EigenLayer have distinct roles, incentivizing different aspects and audiences. Specifically, EigenLayer operates with the goal of bootstrapping new AVS leveraging a subset of the Ethereum validator set. Conversely, Olas staking extends the opportunities to bootstrap AVS participation to a broad range of users, enabling them to operate according to their preferences, leveraging their own token ecosystems, and engage in running their agents periodically or continuously. EigenLayer re-staking model aims to minimize the capital cost for native ETH stakers opting into EigenLayer, avoiding additional cost overhead with an additional token. Simultaneously, Olas Staking works to reduce entry barriers, simplify capital requirements for bootstrapping and broaden operators' participation without the necessity of owning ETH.

Lastly, EigenLayer provides additional revenue streams for ETH stakers, enhancing the Ethereum ecosystem's network effects. Conversely, Olas staking empowers protocols and DAOs across various chains, to boost the liquidity and adoption of their token, fostering more use cases and heightened demand, ultimately fortifying the corresponding protocol or DAO ecosystem.

Conclusion

In this paper, we have presented a comprehensive overview of the Proof of Active Agent (PoAA) mechanism, highlighting its effectiveness in spawning autonomous (AI) agent economies, driving ecosystem growth, and allowing anyone to run said agents and thereby earn rewards. We outline how the mechanism does this by incentivizing agents to achieve predefined Key Performance Indicators (KPIs). We have demonstrated how this mechanism is activated through an innovative staking approach, particularly focusing on cases where on-chain agents activity is verifiable through smart contracts. Notably, we have highlighted the potential for any entity to utilize the same mechanism and architecture proposed for Olas to spawn agent economies aligned with their projects and goals.

For the Olas DAO, members can influence and direct OLAS emissions among staking programs to incentivize specific autonomous services deployed on different blockchains, via a Voting-Weight mechanism.

Already, Olas Predict serves as a practical instantiation of the PoAA mechanism, illustrating how it incentivizes intelligent trades and gives rise to diverse prediction markets. Staking campaigns initiated by Valory under Staker Expeditions further showcase the real-world application of Olas Staking, with active participation from traders providing slashable stakes, demonstrating their contribution to the ecosystem's growth and contributing to over 10% of Safe transactions on Gnosis Chain which were made by Olas-powered agents at the time of writing.

We also highlighted several areas where the PoAA mechanism could be expanded upon in future, including liquid staking, and made a comparison with restaking.
In conclusion, the PoAA, instantiated on Olas as the Olas Staking Mechanism, represents a significant advancement in decentralized economic coordination, by aligning incentives, encouraging active participation of agents, and embracing a multi-chain ecosystem. PoAA sets the stage for the arrival of autonomous (AI) agent economies. Any chain that integrates Olas can benefit from agents becoming daily active users, enriching their protocols and ecosystems.

Appendix

Compatible Staking Contract Configuration

Compatible staking contracts introduced in Staking Contract have the following configuration:

- Token for Rewards: The token in which staking rewards are distributed to participants.
- Number of Agent Instances per Autonomous Service: The predetermined quantity of agent instances associated with an autonomous service registered in the staking contract.
- Max Number of Stakers: The maximum amount of stakers the contract permits at any given time.
- Fixed Amount of Security Deposit and Bond[13]: The established value of non-slashable security deposit and slashable bond required for staking.
- Activity Period: The designated time frame during which the staking contract assesses the activity of the service. This sets a regular interval for evaluating the service's activity, contributing to the determination of staking rewards.
- Max Allowed Inactivity: The maximum duration of inactivity permitted for the service before facing potential eviction. This sets a threshold for acceptable inactivity, preventing services from remaining inactive for extended periods.
- Min Staking Duration: The minimum required duration for which a service must remain staked to qualify for staking rewards. This ensures incentive compatibility, as staking without activity incurs an opportunity cost equivalent to foregone staking rewards.
- Reward Per Second: The designed yield that service can receive per second, contingent on the activity check confirming correct agent activity. The annualized amount represents the active staker's APY.

Reward System Contracts Architecture

The Reward system is a robust and decentralized framework for the allocation of OLAS emissions to staking programs across multiple networks, extending beyond the Ethereum mainnet to chains like Gnosis, Polygon, Arbitrum, Solana, and others. This approach involves the creation of several smart contracts, summarized in what follows. For a visual representation of the interaction of these smart contracts see Fig. 10.

[13] In the current design of compatible staking contracts, the security deposit and bond are denominated in the same token as the rewards.

Tokenomics Contract[14]: Deployed on Ethereum Mainnet, this contract contains information regarding the portion of inflation reserved for whitelisted staking programs per epoch, denoted with maxStakingFraction, and the share of inflation reserved for whitelisted staking programs per epoch, including leftovers from the previous epoch, referred to as effectiveStaking. It is moreover responsible for calculations related to staking reward distributions for implicitly[15] whitelisted staking programs.

VotingWeights Contract: Located on Ethereum mainnet, this contract enables DAO members to assign vote-weights using veOLAS voting power to staking programs eligible for OLAS emissions. It triggers cross-chain messages to broadcast whitelist information.

StakingDispenser Contract:
This pausable contract, deployed on Ethereum:

- enables cross-chain messages to broadcast allocation calculation for staking rewards, ensuring synchronization with the TargetStakingDispenser.
- Allows anyone to mint OLAS corresponding to the staking rewards calculated during the last settled epoch and bridge such amounts to the TargetStakingDispenser via a specified bridge provider.

TargetStakingDispenser Contract: Deployed on a target chain where staking contracts reside, this contract verifies the staking contracts implementation and checks that the staking contract APY is under the threshold provided by the DAO.
If either of these conditions is not met for a specific staking contract, the corresponding reserved OLAS emissions for the epoch are not distributed to that contract and become owned by the protocol. Conversely, the TargetStakingDispenser contract enables the deposit of staking emissions to verified and correctly configured staking contracts based on weights. Additionally, the protocol executor contract on the target chain can drain any remaining emissions in the contract not distributed to staking contracts not deployed via the Staking Factory or not meeting the staking APY threshold (cf. section Staking Contracts Factory).

[14] This requires changes to the currently deployed Olas Tokenomics contract.
[15] See Staking Reward System to understand what is meant with implicit whitelist.