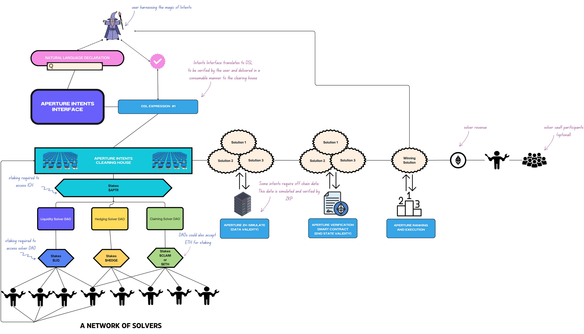
Aperture's Litepaper

An Intent Based Infrastructure powered by a Chatbot UX that revolutionizes the user blockchain interaction

Aperture Finance · Follow
6 min read · Feb 20, 2024

This post serves as an introductory "litepaper" for what's to come at Aperture and for DeFi as a whole. Our mandate at Aperture is to challenge and ultimately transcend the conventional transactional methodologies that have hindered the widespread adoption of DeFi. Herein, we offer a glimpse into the Intents based future.



The TL;DR: 10X better execution for 10X less work

Aperture is building a novel chatbot UX powered by an underlying Intents infrastructure that will enable users to "declare their goals" in natural language and tap a network of solvers to receive better execution and better pricing than what is possible under the current transactional paradigm.

The rest of this post covers several topics.

- UX
- Intents Infrastructure
- Application Layer
- Types of Solvers
- An end to end example

The UX: Aperture LLM powered by an Intents DSL

To begin we'll start with the cornerstone of any mass adoption movement: the user experience (UX). As we discussed in our previous post the current DeFi UX is centered around a *transactional approach* that requires users of varying technical acumen to sign off on a delta or state change hoping the change yields the "end state" they had in mind. With Intents we put that "end state" front and center as the focal point of the user experience.

By using the power of modern LLMs and a proprietary Intents oriented programming language we seek to enhance the expressibility of user Intents. This will allow users to articulate their transactional goals and preferences more intuitively and efficiently, thereby tapping into blockchain's potential with greater ease and precision (which is hopefully a boon for greater adoption).

The best approach to allow a user to declare their desired end state is to remove the constraints of "knobs and buttons" and allow the user to express its natural language what they want. Of course this begets a need to then translate natural language into blockchain code -- this is where the DSL or Domain-Specific Language comes in.

A Domain-Specific Language (DSL) is a specialized computer language tailored to a particular application domain, distinguishing it from General-Purpose Languages (GPLs) that are applicable across a broad spectrum of domains. The design and utilization of DSLs are integral to domain engineering, often involving the creation of new DSLs or the adaptation of existing ones to express problems and solutions more effectively within a specific domain.

In Aperture, the DSL is crafted with a focus on human readability, crucial for supporting clear and intuitive intent expression. This approach differs from other DSLs that may prioritize aspects like programmatic efficiency or machine-level optimization.

On Aperture the LLM bridges the gap between technical functionality and user-friendly interfaces by allowing the user to express in natural language their intent and having that same intent mirrored back to them in a highly readable DSL that can then be served up to solvers as the "declaration of truth" from this user.

Let's use a real world analogy: The LLM to DSL translation UX is similar to a customer placing an order at a pizza restaurant over the phone. The customer might order in very colloquial language -- "Give me your pizza with all the meats in whatever your largest size is". The operator on the other end might mirror back to them "Do you want our Meat Lover's pizza in an Extra Large?". The user easily understands this conversion and agrees -- "Yes, I didn't know the name for it but that's the one I want."

On-chain this interaction would play out in a similar manner. The user might start by declaring their end goal --

"Can you rebalance my ETH-GMX LPs across all my EVM chains to be 80% concentrated on the highest performing setup and the remaining 20% of my LP capital to be evenly split across the remaining pools?"

The DSL translation might mirror back to the user --

- **Eligible assets:** ETH-GMX pairs on Mainnet, Arbitrum, and Avalanche
- **Actions Permitted:** Bridge, Remove liquidity, Swap ETH or GMX, Add Liquidity
- **End Goal 1:** Rebalance liquidity positions to concentrate 80% of **eligible asset** capital on the position with highest spot APR based on data from API Vision.
- **End Goal 2:** Rebalance liquidity positions to concentrate 20% of **eligible asset** capital across the existing pools not used in End Goal 1.
- **Sign** *Intent declaration if correct*

The LLM conversion will codify colloquial language into the standardized terminology of the DSL which can be leveraged by solvers in a predictable and replicable manner.

The Underlying Infrastructure

The Intents-Infra can be broken down into several components:

- **Intents Clearing House (Mempool):** this serves as a preliminary staging area for user intents. It is designed to efficiently organize and queue these intents for processing, using algorithms that prioritize based on various criteria such as urgency and resource requirements. The Clearing House ensures the secure and orderly management of intents before they are committed to the blockchain.
- **ZK-Simulate for Data Validity:** this is a required resource to validate certain intents and corresponding solutions that will rely on off-chain data. A zero knowledge proof can be used to verify the validity of this data. Utilizing advanced cryptographic tools such as Brevis or Axiom, Aperture can generate ZKPs for historical on-chain data that form part of the solver's proposed solution. This method allows for the rigorous verification of the solver's outputs, ensuring that they are accurate, complete, and comply with the specified constraints and intents, without compromising the confidentiality of the transaction data.
- **Verification Smart Contracts:** each type of intent use case will require a smart contract to simulate, verify, and police the proposed solution.
- **Ranking and Execution Engine:** each group of verified intents will need to be ranked based on outcomes and the solver score and then subsequently executed. A crucial aspect of this execution engine is its ability to enforce accountability. Should any nefarious activities occur, such as reverted transactions or other malicious events, the execution engine is designed to penalize the responsible solvers through slashing or other means. This not only safeguards the integrity of transactions but also deters potential malicious behavior by solvers.

The Application Layer: Solver DAOs

The Solver DAO Network is a unique application layer built on top of the Intents-Infra. The Aperture Intents-Infra enables Solver DAOs to focus on enabling and solving unique Intents based use cases without having to worry about the underlying executional needs.

The relationship between Solver DAOs and Aperture

Solver DAOs gain access to the user Intents found within Aperture's Clearing House by staking a requisite amount of $APTR and $ETH1. A Solver DAO can correlate to one large professional solver with a proprietary solution or a network of smaller Solvers.

New Intent solutions can come from Aperture or a third party Solver DAO. Solver DAOs add value by enabling the new Intent use case. This would require submitting the necessary business logic to fit into Aperture's modular design. Once this has been built the use case is now "declarable" from the Aperture Intents Interface or a 3rd party interface spun up by the Solver DAO.

The Aperture DAO will provide $APTR grant assistance to Solver DAO's looking to enable new Intent use cases.

How will Solvers compete and what types of Solver's might exist?

On-Chain vs Off-Chain

In the competitive Aperture Intents ecosystem, Solvers differentiate themselves by the methods they employ to post solutions. While not mandatory, smart contracts are preferred for their scalability and speed. However, off-chain scripts are equally adept at quickly posting solutions, offering an alternative route. Certain declared Intents might even have characteristics that would enable solvers to manually submit solutions (e.g. a seller looking to arrange a large OTC transaction with a 2 day bidding window).

Maintaining Alpha

For Solvers with true "alpha" or proprietary solution-generating methods, they can avoid utilizing a SC to generate their solutions and can instead rely on Aperture's ZK Verification process to build trust around their off-chain script-enabled solution. This will bolster the flywheel effect for onboarding solvers (sustainable business solutions, attracts more revenue, attracts more solvers).

Solver Vaults

Although not explicitly required a Solver in a given ecosystem could also choose to crowd fund their staking requirement via a vault mechanism in exchange for a rev-share on their solver proceeds. Each Solver DAO can open source a rev-share vault contract for their solvers to implement (should they desire bootstrapped funding).

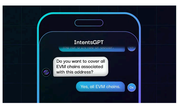
An Example: Airdrop Claim Intent

To make things more digestible let's take the example of the "airdrop claim intent" proposed in our first blog post. How would a user a declare a claiming intent? How could a Solver DAO specializing in claiming tap Aperture's Solver DAO Marketplace?

The user would start by declaring in natural language:

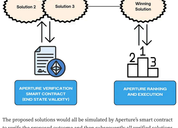
> *"Claim all eligible airdrops on my behalf, including gas cost associated with claiming, in exchange for a 1% or less finder's fee."*

The chatbot might ask clarifying questions to further tease out the user's declaration.



Once this clarification is done the intent would be translated from natural language to the codified Intents DSL and sent back to the user in a readable format for them to verify. From here the Intent expression will be posted into the Intents Clearing House where any eligible Solvers could view the user's declaration.

Any Solver could now view the user's address and cross-reference it with any claimable airdrops or rewards that are eligible for said address. A **permit function powered by an account abstraction wallet** would allow for Solvers to claim any airdrops on the user's behalf. Solvers would compete amongst themselves on the "finders fee" and their overall knowledge of airdrops. Now -- the beauty of Intents, there could be multiple solutions that win out and are executed if Solver A covers a Dymension airdrop and Solver B covers a Celestia airdrop then both Solvers could win a finder's fee from our user.



The proposed solutions would all be simulated by Aperture's smart contract to verify the proposed outcome and then subsequently all verified solutions would be ranked. Aperture would then execute on the user's behalf returning all airdrops!

Llm Defi Intents Tge