Client
API3

Delivered
December 14, 2020

Requirements Specification

Functional Requirements

Requirement	**Details**
DAO functionality is constrained by time periods called "epochs"	
API3 Token holders can stake tokens	Staked tokens enter a liquidity pool that can be used to support a future insurance system Tokens remain staked until token holders unstake them Token holders must submit a request to unstake a specific number of tokens. The request will be honored if abuse-prevention conditions are met.
API3 Token holders cannot abuse the staking system to gain staking rewards without providing commensurate liquidity	Staked tokens can only be withdrawn after a governable cooldown period (unpoolWaitingPeriod) After requesting to unstake tokens, token holders must wait for a governable cooldown period (unpoolRequestCooldown) to elapse before attempting to unstake tokens again
Time-locked tokens can be transferred to the staking pool	Time-locked tokens are preserved if the API3 pool contract is upgraded
Token holders receive inflationary rewards for staked tokens	Inflationary rewards are granted and staked continually, on a per-epoch basis, without intervention from token holders Inflationary rewards are time-locked Time-locked inflationary rewards are preserved if contracts are upgraded

	The inflation rate automatically adjusts in accordance with the parameterized function specified in InflationPrototype.sol.
The DAO contains the foundations necessary to implement an insurance system in the future, as described in the API3 White Paper.	Authorized users of the Oracle contract can register claims to an amount of staked tokens with the DAO An unresolved claim freezes staked tokens as collateral in proportion to the distribution of staked tokens Stakers who wish to withdraw their frozen tokens are instead issued an IOU, which can be redeemed after the claim is resolved. The redeemed amount is equal to the full amount of frozen tokens if the claim fails, or zero tokens if the claim is approved. Stakers receive an IOU if they stake tokens while there is an active claim. In this case, the redeemed amount is equal to zero if the claim fails or to their proportion of the staking pool if the claim is approved.
API3 Token holders can submit proposals to govern DAO functions, fund grants, and execute arbitrary transactions	Submitting a proposal requires a minimum amount of voting power Token holders are limited in the number of proposals they can make per epoch Governable state includes: ● Staking pool parameters ○ Waiting period between requests to unstake tokens ○ Waiting period after staking, before which tokens cannot be unstaked ○ Duration for which inflationary rewards are time-locked ● Inflation rate parameters ○ Minimum inflation APR ○ Maximum inflation APR ○ Staking target (number of tokens) ○ Inflation function coefficient ● Update oracle registry

	• Update identity registry
API3 Token holders can vote on proposals	Voting power directly corresponds to the distribution of API3 tokens Token holders can delegate their voting power to another address
Revenue the DAO receives from API services is burned	
Users can learn the name and URL associated with a participating Oracle by querying a function.	The DAO holds a registry mapping oracle ID numbers to Oracle name-URL pairs
Users can learn the human name associated with a DAO participant's address by querying a function.	The DAO holds a registry mapping Ethereum addresses to human names. This human identity registry is secure from attacks, such as use of fake identities

Non-Functional Requirements

Requirement	**Details**
DAO smart contracts are secure from known smart contract vulnerabilities	Smart contract security is audited and approved by a third party
DAO smart contracts are upgradeable	
Users can interact with the DAO from their browsers using a web application with a graphical user interface	API3 Token holders can submit proposals API3 Token holders can vote on proposals Users can view information related to the DAO's staking pool
The UI displays proposal details so that all voters will be able to verify the specifications of arbitrary transaction proposals	The specifications are hosted on IPFS Details include: • Description • Proposal number • Target contract address • Target function

	● Value (Ether sent) ● Function Parameters
The web application is hosted using decentralized infrastructure	The web application is accessible at independent ENS address The web application is hosted on a decentralized file server (IPFS?)
All code is legally protected by a common open source license	

Client
API3

Delivered
December 14, 2020

API3 DAO Development Roadmap

Item	**Notes**
I. DevOps and user interface	Set up testing environment, starting from LID DAO Aragon template Build user interface to interact with API3 DAO smart contracts
II. Update and refactor API3 Pool contracts to reflect new tokenomics with Better staking UX	Rewards vest as in Timelock Manager
III. Implement automatically-adjusting inflation rate	Discard front-loaded inflationary schedule Prototype to follow
IV. Implement voting and proposal functions and constraints	

V. Implement registries	
VI. Set up Aragon template and voting application proxy contract	LID DAO Aragon template example LID DAO Aragon proxy contract example
VII. Web3 integration: Set up Aragon Connect Intents for proposals	Proposal types: ● Fund grants ● Adjust staking time lock requirements ● Set staking inflation rate parameters ● Update registries ● Create new Aragon roles and permissions ● Upgrade contracts ● Execute arbitrary transactions Incorporate proposal functionality into UI, ensuring that users can initiate proposals and verify transaction details
VIII. Implement insurance claim-related functionality	Register and validate claims: custom Aragon app? Freeze currently pooled funds Switch to issuing IOUs for pool withdrawal and deposit until claim resolves Calculate redemption value of withdrawal IOUs in case of a successful claim and loss of collateral