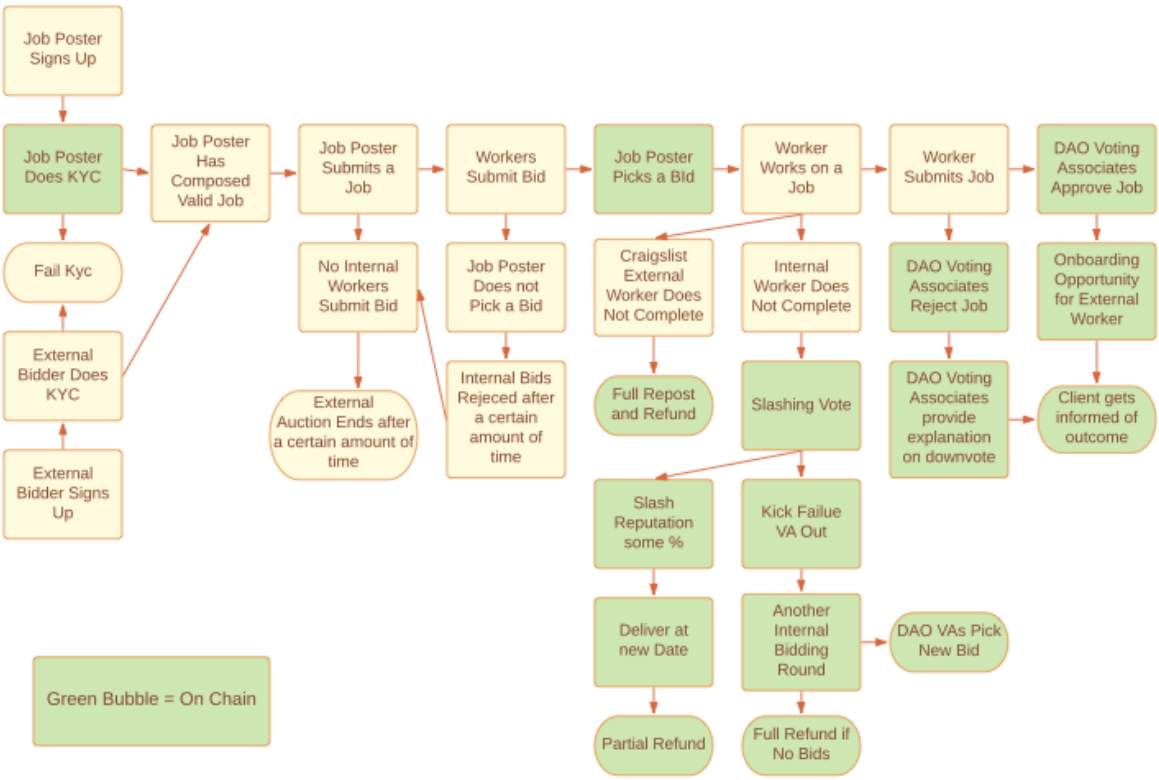
December 1st 2021

OP: Wulf Kaal
Team: David Tai (lead), Robert Carbone,

Summary

This document scopes out both a UX (currently in development) and the following set of on-chain components for a portable, on-chain casper smart contract based DAO. Over the course of the original scoping grant, we have determined what parts are best translated to on-chain smart contracts, and what parts should remain off chain. We chose to focus on core features such as voting for these on-chain components because they are the most mission critical pieces and benefit from being stored as immutable records on the blockchain.

The core features that we refer to are the software that supports the following mission critical flow:



The on-chain components noted in the flow above are highlighted in green and include:

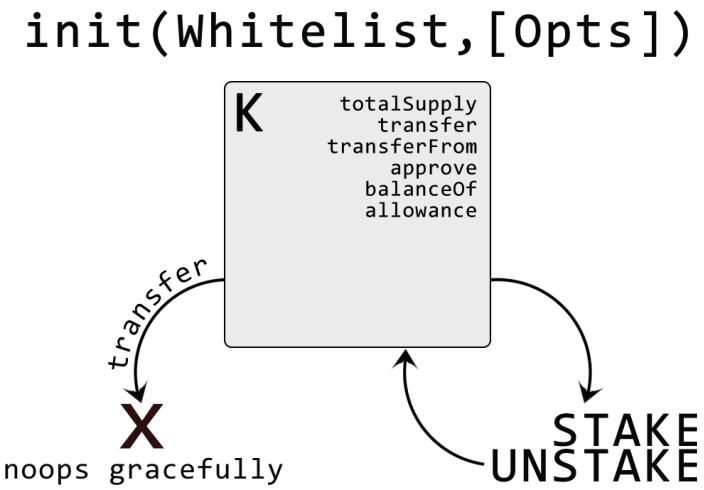
1. KYC
2. Reputation
3. Payment/Escrow/Distribution
4. Voting
5. Platform Variables

Off-chain features currently being developed by Ekontech include:

1. A Reddit style discussion forum that has discussion but not binding decision making power.
2. Sign up & KYC forms need to be centralized on the website but the results will be published to the blockchain as a hash of submitted documents.
3. The bidding process will be handled off chain until a bid is selected, then a new contract is deployed to handle payments, refunds, escrow, etc.

Working document for Service DAO requirements are described here.

We also want to include a grant for looking at the needs of the community as many people have expressed interest in migrating existing DAOs or DAOs with different government models and features onto the casper blockchain



Grant #1 - Non-Transferrable Reputation with Minting Whitelist

This is a CASP-20(ERC-20) based contract that does not support transfer but has burn and mint functionality available from white listed contracts.

1) This contract has an init() function that is single use and supplies the white listed list of contracts that can mint and burn these tokens.
2) This contract's transfer functions noop gracefully or error clearly (define in CEP submission).
3) This contract needs to have the concept of staking/unstaking associated to specific voteIds supplied by the voting contracts and getters to find how many reputation tokens are staked for each user for each vote.
4) This contract has all the CASP-20 spec functions to interoperate with CASP-20 compatible wallets (Gosuto, etc).
5) The constructor can take an optional contract address that allows us to clone the values from an existing reputation repository.
6) Write & submit a CEP proposal.

RFP Input Fields:

Type of Job: Non-Transferrable Reputation with Minting Whitelist

Job Description:

Build a contract that does not support transfer but has burn and mint functionality available from white listed contracts. Modify existing ERC20 library. Create whitelisting system. Integration test whitelist system. Staking features and testing.

Total Price for Job: Euro 40,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tag: Rust smart contracts

Grant #2 - Variable Repository

This is a Key/Value store contract that supports writes from white listed contracts only, its contents are variables that influence voting, bidding, and other contracts.

1) This contract has an init() function that is single use and supplies the white listed list of contracts that can update this Key/Value store
2) The constructor can take an optional contract address that allows us to clone the values from an existing variable repository.
3) Write & submit a CEP proposal.

RFP Input Fields:

Type of Job: Variable Repository

Job Description:

Build a contract that supports writes from white listed contracts only, its contents are variables that influence voting, bidding, and other contracts. Take key value store and the tutorials and modify as whitelist so only certain voting contracts can manipulate it.

Total Price for Job: Euro 20,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tag: Rust smart contracts

Grant #3 - Basic Voting Contract

Reputation and variable repositories can be updated from one or more whitelisted voting contracts. Voting contracts have a list of supported vote types that have specific well defined on-chain actions that will update an associated variable or reputation repository contract.

1) This contract has a function to set the reputation repository contract address.
2) This contract has a function to set the variable repository contract address.
3) This contract contains the logic for the types of votes that mints and burns reputation.
4) This contract contains the logic for updating various variables in the variable repository.
5) Voting Process
    a) This contract has a function called startInformalVote() that takes a timestamp, the type of vote, and voting metadata to start that particular type of vote and returns a ID for the vote
    b) This contract has a informalVote() function that allows account addresses with reputation in the reputation contract to stake reputation in the vote up to their total reputation amount.
    c) This contract has a function called startFormalVote() that takes a timestamp and the voteID generated by startInformalVote, the type of vote, and voting metadata to start that particular type of vote. This can only be called after an informal vote has expired.
    d) This contract has a formalVote() function that allows account addresses with reputation in the reputation contract and have voted in informal to stake

reputation in the vote up to their total reputation amount. The staked values are then updated in the voting repository as this is the binding phase of the vote.

e) This contract has an apply() function that allows anyone with reputation to execute the results of a vote after that vote's timestamp has expired. This will trigger the logic for the vote.

6) There will be a reference list of different types of votes and their pass/fail logic provided.
    a) 2 Different Rounds of Voting for Each Type of Vote
        i) Informal vote
            (1) Does not actually stake reputation or lock anything up in the actual reputation repository
        ii) Formal vote (minting of reputation)
            (1) These need to reference an associated informal vote
            (2) These do stake/lock reputation in the reputation repository until the vote is completed
    b) Types of Votes
        i) Governance / simple vote (no reputation minted)
            (1) These have no side effects on chain
        ii) Admin vote (no reputation minted)
            (1) These set variables in the system
        iii) Project Completion + No Onboarding Vote
            (1) Requires a Bid Escrow Contract address
            (2) Can only be initiated by the job doer on the bid escrow contract
        iv) Project Completion + Onboarding Vote
            (1) Requires a Bid Escrow Contract address
            (2) Can only be initiated by the job doer on the bid escrow contract
        v) Project Non-Completion Refund/Slashing Vote
            (1) Requires a Bid Escrow Contract address
            (2) Can be initiated by anyone with reputation in the reputation repository contract and can only be done after the bid escrow contract's expected date of completion passes (system will make sure there's not an associated project completion vote that's active)
7) Write & submit a CEP proposal.

RFP Input Fields:

Type of Job: Basic Voting Contract

Job Description:

Build a contract that enables variable repositories which can be updated from one or more whitelisted voting contracts. Voting contracts have a list of supported vote types that have

specific well defined on-chain actions that will update an associated variable or reputation repository contract. Enumerate all the different types of votes and functionality. Functionality for onboarding VAs, changing settings, offboarding people. Start with current state and transform into multiple different states to be able to change it throughout by comply with initial state. Minting reputation, redistributing reputation, paying people etc. will all be covered. Use levels of abstraction to reuse different types of code and extend the system. Use Rust crate with an experienced group.

Total Price for Job: Euro 90,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tag: Rust smart contracts

Grant #4 - Bid Escrow Contract

This contract is used for the job approval vote type. A contract is deployed per job approval vote after bidding is completed. The job proposers send tokens to this contract address for this contract to hold. This contract whitelists a vote contract that will control the completion of the transaction as well as take DAO fees.

1) This contract is deployed after bidding ends on a job (currently off chain).
2) This contract's constructor specifies the account address that will be paid out, the contract address that refunds will be sent to, the voting contract that will control this escrow process, expected date of completion, and contain the CASP-20 contract addresses that are associated with payout tokens.
3) Payment Process
    a) This contract waits to receive funds from a user.
    b) Once funds are deposited in this contract, a vote can be created to transfer the tokens using the whitelisted voting contracts.
    c) This contract is only aware of the agreed upon CASP-20 tokens that it is initialized with and will ignore all others.
    d) The voting contract can send the total token amounts to both the job doer and the people with reputation in the system after the vote executes. Or it can refund if there's a refund vote.
    e) The escrow contract pays proportional to the reputation score of each reputation holder after the formal vote is final

RFP Input Fields:

Type of Job: Bid Escrow Contract

Job Description:

Build a contract that is used for the job approval vote type. A contract is deployed per job approval vote after bidding is completed. The job proposers send tokens to this contract address for this contract to hold. This contract whitelists a vote contract that will control the completion of the transaction as well as take DAO fees. Fund contract and allow specific voting contract to control the sending of the money. Distribution system would be on the voting side. It would have the ability to send the tokens in the wallet to multiple recipients. Voting engine would determine the split of the assets. Send to all the people with reputation versus the person who did the job. Simple public airdrop contract would need to be created on Casper.

Total Price for Job: Euro 20,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tag: Rust smart contracts

Grant #5 - KYC Contract

This contract stores hashed KYC information associated with account addresses in the system. This is for book keeping requirements to be used in the process of onboarding job proposers and voting associates.

1) This contract stores a hash of the KYC data used to add users to the system to use the chain as a witness system.
2) Write & submit a CEP proposal.

RFP Input Fields:

Type of Job: KYC Contract

Job Description:

Build a contract that stores hashed KYC information associated with account addresses in the system. This is for book keeping requirements to be used in the process of onboarding job proposers and voting associates. Simpler version of variable depository from grant #2.

Total Price for Job: Euro 10,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tag: Rust smart contracts

Grant #6 - Make the Off Chain UX Work OnChain

support for the on-chain contracts in the UX for the offchain version of the DAO software

RFP Input Fields:

Type of Job: UX buildout

Job Description:

Support for the on-chain contracts in the UX for the offchain version of the DAO software.

Total Price for Job: Euro 20,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tags: UX code

Grant #7 - DAO Datatype Analysis

RFP Input Fields:

Type of Job: Data Analysis and Specs

Job Description:

Analyze the data types and their uses for the DxD onchain transition

Total Price for Job: Euro 14,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tags: Research

Grant #8 - Future DAO GAP Analysis

We will create a follow up grant for features that the community want added to the DAO or are present in other operational DAOs that we want to bring over to our software.

1) Perform a GAP Analysis on missing features between the DAO software and other DAO models to discover common features that we could add to the software.
    a) In addition, we've already identified certain key features that the community finds interesting but are not included in the base DAO software
        i) Add the concept of milestones
        ii) Different voting mechanisms
        iii) Add surveys
        iv) Support mandatory DOS fees
        v) Different payout mechanisms

RFP Input Fields:

Type of Job: research on GAP analyses for future DAOs on-chain on Casper

Job Description:

Perform a GAP Analysis on missing features between the DAO software and other DAO models to discover common features that we could add to the software.

Total Price for Job: Euro 20,000.00

Job Start Date: January 30th, 2022

Job Completion Date: April 30th, 2022

Tags: Research

Grant #9 - DxD Database Transition On-Chain

RFP Input Fields:

Type of Job: Database transfer on-chain on Casper

Job Description:

Define data types and move all agreed upon data onchain on Casper. Take #3 grant smart contracts - explore the usability and transition existing DxD database. Figure out which transactions fail. Move everything working properly with redundancy. Testing for all functionality. 2 engineers for 2 weeks full plus project management. Add 70 hours for troubleshooting.

Total Price for Job: Euro 50,000.00

Job Start Date: April 30th, 2022

Job Completion Date: May 30th, 2022

Tags: Rust smart contracts

Captability for DAO to own another contract

Post a new piece of code with proposed upgrade - DAO would vote for upgrade.

Voting reputation tokens should own the contract

Who installs contracts and who upgrades the contract

On ethereum it goes through Aragon

EAch contract has an owner and owner can modify the contract. People vote with multisig call. On casper this problem is not solved yet. No proper ownership model.

How different CurveDAO Olympus DAO etc. Top 10 DAOs - they usually have multiple voting and escrow contract. EAch of these contracts has an owner which is another contract. Currently only accounts can own contracts in Casper.

Example Uniswap token holders should own the uniswap contract

Need separate grant for ownership of contracts on Casper

Currently the closes thing to the DAO model may be Gitcoin

Create one account with 10 people and 9 of them sign the contract upgrade.

Only uniswap and maybe two other projects on Casper use tokens in another contract.

Separate research grant - token governed DAO contract