ADADAO Specification

ADADAO is a DeFi protocol on Cardano blockchain which facilitates creation of fully collateralized stable coins leveraging Native Cardano Assets.

Scripts observation

There are 2 types of scripts in the ADADAO protocol:

- **Single-UTXO Scripts:** Oracle, Management, Protocol, Stablecoin, Stability Pool, Liquidation, Governance, AdaoPool, PSM scripts contain only one UTXO, which stores service information in the datum.

- **Multi-UTXO Scripts:** Borrowing, Proposal, StakingPool scripts consist of multiple UTXOs belonging to different users.

All scripts can be initiated only by the manager with the corresponding initial configuration. Any changes in datum and redeemer structure, as well as in validator checks, can be added through migration and only by the manager. The information stored in the script before migration will be converted to the new format during migration and should not be lost in the new script version. The conversion and data transfer to the new script should be double-checked by developers and testers who made the changes to the script.

ADADAO contracts

Oracle

Oracle is a service used for delivering real-world values to the blockchain. In ADADAO, the Oracle serves as a source of the `lovelacePerCent` exchange rate, which is utilized to calculate an important system parameter - the `collateral to debt` ratio. Additionally, the fixed time in the Oracle datum is used to verify both stake creation and withdrawal dates.

The Oracle contract supports 2 actions:

- **InitFeed:**

This action creates an Oracle script with the actual lovelacePerCent exchange rate and a datetime label. It is called from the Oracle manager's key wallet upon project startup.

- **UpdateFeed:**

This action updates the lovelacePerCent exchange rate and datetime label. It is called periodically from the Oracle manager's key wallet, with the time period predefined in the automatic action worker.

The current exchange rate provider for the system is the CoinGecko API. The value of lovelacePerCent is retrieved from the CoinGecko API and then stored in the ADADAO Oracle along with a timestamp.

Management contract

The Management contract serves as a core component of the system, responsible for initiating and managing all services within the ADADAO.

Management Contract Functions:

- **Start management script:**

Initiates the Management script, thereby fixing all system information in the blockchain. Once started, key system parameters, such as the manager of the current ADADAO system, are immutable for security reasons. This prevents unauthorized creation or migration of service scripts from unknown wallets.

- **Start service script:**

Used for the initial launch of all other scripts within the system, excluding the Oracle, which operates as a separate service independent of the ADADAO protocol. This function facilitates the addition of new scripts to the ADADAO project using the same configuration as existing scripts.

- **Migrate management script:**

Enables the upgrade of any system script version, excluding Management itself and the Oracle. This functionality becomes necessary in the event of script changes, such as alterations to datum structure, redeemer, or validator checks. Migration can only be initiated by the manager.

- **Provide Manager Information:**

Supplies manager information to contracts utilizing managerAddress/managerPkh for signature verification or to validate transfers to the manager's wallet.

Protocol contract

The Protocol script contains the UTXO (Unspent Transaction Output) with the configuration of the ADADAO system.

Configuration parameters

- `minCollateralBalance`: Minimum collateral required in the Borrowing protocol.
- `minDebtBalanceAusdCents`: Minimum debt required in the Borrowing protocol.
- `individualCollateralToDebt`: Collateral to debt ratio for calculating allowed debt from deposited collateral.
- `liquidationRatio`: Collateral to debt ratio required to initiate vault liquidation.
- `stabilityFee`: Fee paid once by borrowing AUSD.
- `minAUSDCentsToInvest`: Minimum AUSD cents allowed to invest in the Stability Pool.
- `stakingRate`: Annual rate for staking ADAO.
- `oracleConfiguration`: Configuration for the lovelaceToCents feed oracle.
- `quorum`: Voting amount required to suggest proposal voting eligibility.

- `proposalCost`: Cost of creating a system changes proposal.
- `votingDuration`: Duration of proposal voting.

More information about the usage and updating of configuration parameters is provided in sections about Borrowing, Stability Pool, Liquidation, Staking Pool, and Proposal scripts.

Protocol actions

- **Start protocol:**

Protocol creation is conducted under the validation of the management script. The initial configuration is created from a predefined configuration file. This action can only be initiated from the manager's wallet.

- **Migrate protocol:**

Protocol migration is also conducted under the validation of the management script. The new configuration is obtained from the predefined configuration file. This action can only be initiated from the manager's wallet.

- **Update configuration:**

The configuration can be updated through the DAO (Decentralized Autonomous Organization) proposal service.

- **Vault creation:**

The protocol participates in vault creation to validate the initial vault data. To prove the legality of vault creation under the protocol's control, the vault will contain the VerificationToken in its UTXO.

- **Provide configuration info for reading:**

Used to retrieve the configuration parameters for reading and utilization in validators of other scripts.

Stablecoin contract

The Stablecoin script stores information related to the AUSD (ADADAO Stablecoin) - its currency and token name.

Stablecoin contract actions:

- **Start stablecoin script:**

The creation of the Stablecoin script is initiated under the validation of the management script. The initial datum will contain the currency and token name of the official ADADAO stablecoin - AUSD. This action can only be initiated from the manager's wallet.

- **Migrate stablecoin script:**

Migration of the Stablecoin script is also conducted under the validation of the management script. This action can only be initiated from the manager's wallet.

Note: It is crucial that the Stablecoin currency and token name remain unchanged to prevent inconsistencies in services and AUSD minting policy.

- **Provide AUSD info for reading:**

Used to retrieve the AUSD information for reading and utilization in validators of other scripts.

Reserved actions

Reserved actions are required when changes to the AUSD policy are necessary. In such cases, the stablecoin script should be restarted (a new one should be started, without migration), and the manager should enable the stablecoin exchange to burn old AUSD in exchange for new system AUSD tokens (created from the new policy).

Important! Reserve actions are not fully implemented yet, but validation checks have been added. It is planned to finalize the exchange logic after the mainnet release.

Enable stablecoin exchange: Enables the exchange and fixes newStablecoin parameters in the datum. This action should be called by the manager.

Exchange: This action should be called by the user to burn old stablecoins and receive a receipt for obtaining new stablecoins in exchange.

Borrowing contract

The Borrowing contract enables users to obtain AUSD tokens from the ADADAO stablecoin protocol. To obtain AUSD tokens, a user must first open a vault. Then, the user must deposit Ada collateral into their vault script. Based on the deposited collateral, the system calculates the amount of AUSD available for borrowing (ensuring that the final collateral-to-debt ratio is higher than the CTL parameter set in the Protocol configuration). After borrowing, the collateral amount used to cover the borrowed AUSD remains locked in the script until the borrowed AUSD is repaid to the system. The amount of locked Ada is recalculated each time the system receives new exchange rate information from the Oracle. A user may withdraw the unlocked part of the collateral at any time. To release the collateral (either fully or partially), the user must repay the borrowed AUSD amount (either in full or in part). If the AUSD debt is fully repaid, the user can withdraw the full collateral amount and close the Vault.

If the Ada market price decreases, there is a possibility that the Vault becomes undercollateralized (i.e., the Vault's collateral-to-debt ratio falls below the liquidation ratio set in the Protocol configuration). In this case, the vault's collateral is seized by the system, and the Vault's debt becomes zero. After seizing the collateral, the user is no longer required to repay any AUSD tokens to the system.

Borrowing related actions:

- **Open Vault:**

Checks the initial Vault Datum under the Protocol control and mints a Vault Verification token. It also mints a Vault thread token (ID token) and sends both tokens to the Vault script with a minimal Ada amount and initial Vault Datum. Action can be called by any user.

- **Deposit Collateral:**

Sends an Ada amount from the user to their Vault.

- **Withdraw Collateral:**

Sends an Ada amount from the Vault script to the user. There are several limitations on updating the collateral value, which the protocol checks off-chain and on-chain in the validator:

- The resulting collateral in the Vault should be >= `MinCollateralBalance`, as configured in the protocol.
- The resulting collateral-to-debt ratio should be greater than `IndividualCollateralToDebt`, as configured in the protocol.

- **Borrow:**

Locks a particular Ada amount on the Vault script, mints AUSD tokens, and sends them to the borrower.

- **Repay Debt:**

Burns AUSD tokens received from a user, releasing the collateral amount (either fully or partially) on the Vault script. Limitations for updating the debt value:

Limitations for updating debt value:

- The resulting debt in the Vault should be >= `MinDebtBalance`, as configured in the protocol.
- The resulting collateral-to-debt ratio should be greater than `IndividualCollateralToDebt`, as configured in the protocol.

- **Close Vault:**

Burns the Vault verification token and Vault thread token, and sends a minimal Ada amount to the user (which was placed on the script while opening the vault).

All previous actions were for any user, the next one is available to call only under manager wallet:

- **Seize Collateral:**

Action to seize collateral from the Vault during the liquidation process. More about collateral seizing is in Liquidation doc.

- **Migrate vaults:**

Vaults migration is conducted under the validation of the management script. All vaults data should be transferred to the new borrowing script without changes. This action can only be initiated from the manager's wallet.

Collateral to debt validation

To validate if the resulting collateral-to-debt ratio in the Vault is allowed, we retrieve the `ExchangeRate` from the Oracle and `IndividualCollateralToDebt` from the protocol configuration. Since we measure the collateral in ADA and the debt in AUSD, we convert both to the same units. Assuming that the `ExchangeRate` is 1 AUSD to 1 ADA rate, we can conveniently convert Debt to ADA:

`debtInAda = debtInAusd * exchangeRate`

After that, we can calculate the ratio:

`collateralToDebt = collateralInAda / debtInAda`

Then, we compare it with IndividualCollateralToDebt and forbid the transaction if the predicate below is not satisfied:

`collateralToDebt` > `IndividualCollateralToDebt`

StabilityPool Contract

StabilityPool is a script that contains funds that may be involved in liquidation process and this way provides the system with stability. StabilityPool Funds consist of all the AUSD funds provided by StabilityPool Providers that are used for emergency buying undercollateralized debts, and ADA funds that are received from vaults with undercollateralized debts after seizing collateral. StabilityPool Providers that want to take part in liquidation process invest their AUSD funds to StabilityPool and have back a Receipt token with information about provided funds. When the collateral currency market value decreases and `IndividualCollateralToDebt` ratio for particular vault becomes less than `LiquidationRatio`, liquidation process must be initiated. For every vault with undercollateralized debt the following operations must be done:

- the same AUSD amount as undercollateralized debt value must be taken from StabilityPool and be burned,
- collateral funds locked on Vault script address must be seized and sent to StabilityPool,
- the information about StabilityPool AUSD funds before liquidation, burned AUSD amount and seized collateral amount must be added to Liquidation script Datum.

StabilityPool Datum stores a liquidation counter that indicates how many liquidations were invoked. When a provider decides to receive rewards, the final rewards amount is calculated according to the provider's AUSD investment and liquidations this investment took part. After calculations are finished the provider may receive the final funds from StabilityPool.

Stability pool actions:

Start Stability pool: Stability Pool creation is conducted under the validation of the management script. The initial datum contains the initial liquidation counter (epoch = 1, number = 0), indicating that no liquidations have been performed yet. This action can only be initiated from the manager's wallet.

Migrate Stability Pool: Stability Pool migration is also conducted under the validation of the management script. Liquidation counter and other stability pool data shouldn't be changed during the migration. This action can only be initiated from the manager's wallet.

Deposit AUSD:

Each user who wishes to contribute to the stability of the ADADAO Stablecoin system may invest their AUSD funds into the StabilityPool if the following conditions are met:

- The provider has sufficient AUSD funds in their wallet.

- The invested funds are equal to or greater than the minimal AUSD amount required for investment (a parameter set in the Protocol configuration).

Each provider may invest funds multiple times without limitations. After depositing AUSD funds into the StabilityPool, a new Receipt token is created and transferred to the user. The receipt token contains information about the deposit.

Example:

Receipt token name `r1.0.5000` constructed from:

- r: receipt
- 1: liquidation epoch
- 0: liquidation number
- 5000: AUSD cent amount

This information will be used in ADA rewards calculations.

Receive ADA rewards or full withdraw:

Provider may withdraw funds (all the remaining funds or Ada rewards only) any time according to his Receipt (even if the invested AUSD funds were not fully spent) if the following conditions are complied:

- the receipt was created by Provider himself
- there are enough funds in StabilityPool. Provider can withdraw/receive rewards only one receipt by one action call.

In case, when user receives only rewards without withdrawing AUSD deposit, old receipt should be burned and new one mint with current liquidation epoch and number, to calculate rewards exactly from that moment.

In case, when user withdraws deposit, the receipt token is burning.

AUSD deposit Rewards calculation

Unspent AUSD deposit for every receipt is represented as a Product evaluated according to the following formula: `Product = d * (1 - Q / D)` where d - unspent AUSD amount for current Receipt before the current liquidation Q - total liquidation amount D - total StabilityPool AUSD current funds

ADA reward is represented as a Sum evaluated according to the following formula: `Sum = d * (E / D)` where d - unspent AUSD amount for current Receipt before the current liquidation E - total ADA amount gained by StabilityPool after current liquidation D - total StabilityPool AUSD current funds The function for calculating Sum is called recursively with accumulating results for all the liquidations happened after depositing.

Liquidation contract

Liquidation contract contains of Liquidation script only. It is a script that collects information about all the completed liquidations within the epoch. When the market price of Ada falls, it may happen that some Vaults become undercollateralized. To make AUSD stablecoin really stable, uncovered debts must be liquidated. Liquidation script stores in Datum the current LiquidationCounter and information about completed liquidations. Liquidation Datum may be either empty `InitLiquidationDatum`, which means that no liquidations have been invoked yet, or `LiquidationDatum` with `LiquidationInfo` with the following information:

- `LiquidationCounter` - contains current liquidation information: epoch and liquidation number for the current epoch
- `Map LiquidationCounter LiquidationParameters` - contains pairs of completed liquidations counter and particular parameters which are used for calculating Providers' rewards.

As a script Datum size is limited in the Cardano blockchain we have to limit the size of the LiquidationCounter/LiquidationParameters Map, so the size of the Map is defined as `epochSize` constant. When the liquidation number for the current epoch is bigger than the `epochSize` constant, we start a new epoch with a single entry (new liquidation information) in the Map and LiquidationCounter with incremented epoch (after starting a new epoch the LiquidationCounter is defined as `LiquidationCounter {lcEpoch = succ currentEpoch, lcNumber = 1}`).

The ADADAO system automatically monitors the state of vault collateralization. It periodically recalculates the current CTL of vaults based on the actual exchange rate obtained from the Oracle. If there are candidates for liquidation (vaults with collateralization ratios lower than the liquidation ratio), liquidation is initiated for each vault one by one.

Liquidation process is divided into 2 steps:

1. **seizing collateral:** locked collateral of an unbacked vault is passed from vault script to system, Vault script debt becomes zero.

2. **liquidating unbacked AUSD tokens:** burning AUSD tokens and paying seized collateral amount to AUSD tokens providers. For now only liquidating with StabilityPool approach is implemented.

Liquidation with StabilityPool is divided into 2 steps:

1. burning AUSD tokens taken from StabilityPool and putting seized collateral to StabilityPool,

2. adding liquidation information to Liquidation script Datum (to give StabilityPool providers the information for calculating their rewards)

If StabilityPool doesn't have enough funds to perform the liquidation, the liquidation stays ignored (as we don't have other liquidating ways for now).

Liquidation related actions:

- **Start Liquidation Script:**

Liquidation Script creation is conducted under the validation of the management script. The initial datum is empty `PInitLiquidationDatum` constructor, indicating that no liquidations have been performed yet. This action can only be initiated from the manager's wallet.

- **Migrate Liquidation Script:**

Liquidation Script migration is also conducted under the validation of the management script. Liquidation data shouldn't be changed during the migration. This action can only be initiated from the manager's wallet.

- **Seize Collateral:**

Seizing collateral from undercollateralized vault, minting liquidation token that contains information about seized collateral amount and unbacked AUSD tokens amount, and passing seized collateral and liquidation token to manager wallet.

- **Liquidate With Stability Pool:**

Burning AUSD tokens taken from StabilityPool (AUSD token amount is taken from liquidation token), sending seized Ada collateral from manager's wallet to StabilityPool (Ada amount is taken from StabilityPool), incrementing LiquidationCounter in StabilityPool datum.

- **Update Liquidation Data:**

Adding information about new liquidation to the Liquidation script datum, incrementing Liquidation counter in the Liquidation script datum, burning liquidation token.

Governance contract

The Governance script stores information about the ADAO coin - its currency and token name.

Governance actions:

- **Start Governance:**

Governance creation is conducted under the validation of the management script. The initial configuration is created from a predefined ADAO configuration. This action can only be initiated from the manager's wallet.

- **Migrate governance:**

Governance migration is also conducted under the validation of the management script. This action can only be initiated from the manager's wallet.

- **Proposal creation:**

The governance script participates in proposal creation to validate the initial proposal data. To prove the legality of proposal creation under the governance's control, the proposal will contain the VerificationToken in its UTXO.

- **Provide ADAO info for reading:**

Used to retrieve the ADAO information for reading and utilization in validators of other scripts.

ADAO Pool contract

ADAO Pool is a system script used to store ADAO coins, which will be utilized as staking rewards.

ADAO Pool Actions:

- **Start ADAO Pool:**

ADAO pool creation is conducted under the validation of the management script. This action can only be initiated from the manager's wallet. The initial ADAO balance of pool is 0.

- **Migrate ADAO Pool:**

ADAO Pool migration is also conducted under the validation of the management script. This action can only be initiated from the manager's wallet.

- **TopUp ADAO pool:**

This action is used by protocol owners to top up the pool with ADAO. If the pool is empty, there is no way to receive stake rewards. Anyone who owns ADAO coins may top up the pool.

- **Stake creation:**

The ADAO pool script participates in stake creation to validate the initial stake data. To prove the legality of stake creation under the ADAO pool's control, the stake will contain the VerificationToken in its UTXO.

- **PayRewards:**

Rewards from ADAO staking may be withdrawn by the stake owner at any time (provided that ADAO rewards are >1, and only integer amounts may be withdrawn). The formula for calculating simple interest on a deposit on any withdrawal day without compounding is:

```
Interest = Principal × Rate × (Number of Days / Total Days in a Year)
```

Where:

Interest is the total interest earned, *Principal* is the initial deposit amount, *Rate* is the annual interest rate (expressed as a decimal), *Number of Days* is the number of days the deposit is held, *Total Days* in a Year (365 days)

Proposal contract

The Proposals module is one of the DAO functions in ADADAO. Proposals serve as a mechanism for suggesting configuration changes and conducting decentralized voting on those changes. Voters are ADAO token holders who store them either in wallets or in the ADADAO staking pool.

Proposal actions:

- **Create proposal:**

Proposal creation is conducted under the Governance script validation to check all parameters in the proposal datum. The `quorum`, `cost`, and `deadline` parameters, once calculated from protocol configurations (`quorum`, `proposal cost`, `voting duration`), will be permanent for the created proposal and fixed in its datum.

In ADADAO, there are 2 types of configurations that might be changed:

1. Protocol: The following configuration values might be proposed to change in the protocol:

- MinCollateral
- MinDebtAcent
- IndividualCtl
- LiquidationRatio
- StabilityFee
- MinInvestmentAcent
- StakingRate
- Quorum
- ProposalCost
- VotingDuration

2. PSM For the PSM, users might propose to add or remove stablecoins from the list, enabling or disabling the possibility to mint AUSD with those stablecoins. Additionally, there is an option to disable (or enable if disabled) the PSM module in the ADADAO service as an auxiliary mechanism for AUSD minting.

All proposed configuration values pass through on-chain validation to ensure they have a value from the allowed value segment and that the proposal will indeed change something in the system.

Any user is allowed to propose system configuration changes. The creation of a proposal costs ProposalCost Ada, but if the voting reaches quorum, the proposal initiator will receive that cost back. Paid proposal is a mechanism safeguards the system from unnecessary proposals that could have a negative impact on the community.

- **Vote:**

Once a proposal is created, it awaits votes from DAO keepers until the deadline for voting is reached.

There are two methods for voting for or against a proposal:

1. Using ADAO tokens:

Potential voters should possess ADAO tokens in their wallet and may use them to vote on the proposal, with one token equaling one vote. When a voter casts their vote, it is added to the total votes for or against the proposal, and their ADAO tokens are locked within the proposal's UTXO. In return, the voter receives a Voting Receipt Token that aligns with the proposal's policy, preserving information related to the proposal. The Voting Receipt Token includes details about the vote—either for or against—and the amount of ADAO tokens involved. For example, in a Voting Token named `v0.10`, `v` signifies it as a Voting token, `0` indicates the vote is against (whereas `1` would signify a vote for), and `10` represents the ADAO amount, which equates to the voting amount.

Upon the conclusion of the voting period, voters can reclaim their ADAO tokens using this receipt.

2. Using ADAO stake:

ADAO stakers can participate in voting without losing their stake interest. When a voter casts their vote using their stake, the entire stake is considered as their vote and is locked from withdrawal for the duration of the voting process. Stakers still retain the ability to receive stake interest during this time, but they temporarily cannot withdraw their stake. Once the voting period concludes, the stake is unlocked and can be withdrawn as usual.

- **Process voting results:**

Voting is structured on a quorum-based schema, where the quorum represents the total sum of all votes cast for the proposal (including both for and against votes). The outcome of the voting process is considered valid only if the total number of votes meets or exceeds the specified quorum value.

The process of voting result analysis occurs periodically and is initiated from the manager's wallet. It identifies all proposals that have reached the deadline, analyzes the results, and applies them if they are accepted.

Proposals can have one of three possible outcomes:

1. **Quorum is not reached.** Quorum not reached: This indicates that the community lacks interest in the proposal. As a result, the system rejects the proposal, and the proposal creator cannot receive a refund of the proposal cost.

2. **Quorum reached, result is against:** In this scenario, the majority has voted against the proposal. Consequently, the system rejects the proposal, but the proposal creator can receive a refund of the proposal cost. This outcome also applies when the number of votes against is equal to the number of votes for.

3. **Quorum is reached, result is for.** This signifies that the majority has voted in favor of the proposal. As a result, the system initiates the update of the configuration with the new values. Proposal creator can receive a refund of the proposal cost.

- **Withdraw ADAO votes:**

When the proposal voting result is processed by the system, stake votes are automatically released. However, ADAO votes require user action; users must withdraw their ADAO votes from the proposal using the Vote Receipt Token. Upon withdrawal, the Vote Receipt Token is burned.

It's important to note that there is no way to withdraw votes or release stake from the voting process before the proposal is finished.

- **Close proposal:**

When a proposal is concluded and there are no ADAO votes remaining in the proposal (all voters have withdrawn their ADAO votes), the proposal initiator can close the proposal and receive the proposal cost and script creation cost (minimum ADA value) back. If the quorum is not reached, the user will only receive the script creation cost (minimum ADA value) back, and the proposal cost will be seized in favor of the system.

- **Migrate proposals:**

Proposals migration is conducted under the validation of the management script. All proposals data should be transferred to the new borrowing script without changes. This action can only be initiated from the manager's wallet.

Staking pool contract

Staking pool is a mechanism that allows participants to stake their ADAO tokens and receive an annual percentage. Participants in a staking pool can deposit their tokens into the pool and earn reward based on the size and duration of their deposit. ADAO staking pool works as non-compounding staking. That means that rewards are calculated only on the initial deposit, meaning that accrued rewards do not increase over time. Rewards from ADAO staking can be claimed daily (only whole numbers of ADAO rewards can be withdrawn), not just at the end of the year. Furthermore, staked tokens can be utilized for voting on proposals within the system without interrupting the interest rewards accumulations.

Staking pool actions

- **Stake ADAO tokens:**

When user stake ADAO tokens, new UTXO is creating under the ADAO pool validation control, where the date of stake creation checks to guarantee correct rewards calculation.

- **Withdraw ADAO stake:**

User can withdraw the deposited ADAO tokens and whole amount of ADAO rewards at any time. One limitation is if stake is used in proposal voting, it temporary can't be withdrawn until proposal will be finished. In taht case user might receive only rewards and return to withdraw later.

- **Claim only rewards:**

User can claim only rewards at any time when there is more than 1 ADAO accumulated. Rewards will be paid from the special rewards pool - ADAO pool, that contain reserved ADAO only for that purpose. Currently, in ADADAO, the calculation of staking rewards is based on a year consisting of 365 days.

- **Migrate stakes:**

Stakes migration is conducted under the validation of the management script. All stakes data should be transferred to the new borrowing script without changes. This action can only be initiated from the manager's wallet.

Peg Stability Module (PSM) contract

The Peg Stability Module (PSM) is a component of the Stablecoin Protocol system within the market segment. It enables users to purchase and sell AUSD tokens with a price close to 1:1 (1 AUSD = 1 USD) without requiring collateral with surplus. Additionally, the PSM module contributes to the stability of the system by storing income obtained from selling AUSD tokens as fiat stablecoins.

User's Perspective:

When a user seeks to acquire AUSD tokens, they have two options:

- Utilize the Borrowing service, which involves opening a Vault, providing collateral, borrowing AUSD, and optionally repaying AUSD and withdrawing collateral.
- Engage in swapping Ada for AUSD tokens and vice versa at or near market price directly on the PSM.

The decision on which method to use depends on the user's requirements and the prevailing cryptocurrency landscape. For instance, during periods of rising Ada prices, leveraging the borrowing service can be advantageous as it enables users to preserve Ada funds in the Vault while simultaneously utilizing the acquired AUSD stablecoins. Conversely, when Ada prices decline, it may be more beneficial to swiftly exchange Ada tokens for AUSD stablecoins at or near market price. Thus, both services offer value and utility to users.

AUSD Price Regulation:

Under this approach, the AUSD price is partially pegged to the fiat stablecoin price, and the system does not employ additional price regulation mechanisms. Instead, the regulation of AUSD price is delegated to the regulation of fiat stablecoin prices. As fiat stablecoins are invested in maintaining a USD:USDC(or another fiat stablecoin) price close to 1:1, they employ their own techniques for regulation. Users always have the ability to buy or sell AUSD tokens at market price, provided there are sufficient fiat stablecoins available in the Cardano blockchain for purchase and enough fiat stablecoins in the PSM pool available for sale.

PSM actions:

- **Start PSM:**

PSM creation is conducted under the validation of the management script. The initial datum contains the predefined configuration with enabled or disabled PSM and the allowed stablecoins list. This action can only be initiated from the manager's wallet.

- **Migrate PSM:**

PSM migration is also conducted under the validation of the management script. This action can only be initiated from the manager's wallet.

- **Buy AUSD:**

This action enables users to swap stablecoins allowed in PSM configuration to AUSD. AUSD will be minted, and stablecoins received from the user will be stored in the PSM script.

- **Sell AUSD:**

This action allows users to sell AUSD for stablecoins allowed in PSM configuration. The requirement is that the PSM has enough stablecoins on the script to pay the user. AUSD will be burned as a result of the swap.

- **Update PSM configuration:**

The configuration can be updated through the DAO proposal service.

Enhancement Agenda

Oracle

Implementing an Oracle Pool with multiple providers is recommended to enhance the security and reliability of the system. This feature allows for the comparison of exchange rates obtained from different providers, enabling anomaly detection using statistical methods such as standard deviation.

An alternative approach to enhance reliability is to utilize renowned Oracle providers who can supply the secure Oracle pool with the required values for the system. By leveraging established Oracle providers, there may be no need to develop a custom Oracle pool.

Management

1. Management Wallet Loss Scenario Handling:

Implement robust measures to address the potential loss of the management wallet, ensuring continuity of system management even in such scenarios. This could involve setting up a backup management wallet with appropriate authorization protocols and failover mechanisms to seamlessly transition management responsibilities in the event of a loss.

2. System Freeze Feature:

Introduce a system freeze feature to handle situations where the system becomes temporarily unable to function properly. This feature would enable administrators to halt all system activities temporarily until the underlying issue causing the disruption is resolved. During the freeze period, no transactions or modifications would be allowed, ensuring data integrity and preventing potential errors or security breaches.

Stablecoin

In the event of a change in the AUSD policy, there is currently no mechanism to migrate old AUSD to the new AUSD currency, as can be done with script migration, because AUSD will be in circulation among users, wallets, exchanges, and markets. Therefore, the recommended approach is to start a new stablecoin script from scratch (not through migration) with the new AUSD policy and enable exchange in the old stablecoin script. Thus, it is necessary to implement an exchange contract for users to burn their old AUSD and receive the new AUSD tokens in return. It's important to note that this logic is still in the draft stage within the project, and thus it requires further refinement and extensive testing before deployment.

ADAO Pool

Currently, the staking interest formula does not account for leap days. To accurately consider leap days, it is recommended to incorporate information from the Oracle with the current year to determine if it is a leap year or not. By integrating this data into the staking interest calculation, the ADAO Pool can provide more accurate and reliable rewards to stakeholders, enhancing the overall functionality and fairness of the staking mechanism.

Liquidation

1. Empty stability pool issue:

There may be different methods for performing the liquidation, such as liquidating with the StabilityPool, conducting debt auctions, or redistribution. However, currently, only the liquidation with the StabilityPool approach has been implemented. In the event that the StabilityPool is empty due to various reasons—such as all AUSD in the pool being used up or minimal deposits—the liquidation process will not function, and the system unable to return to a stable state.

2. The liquidation epoch size:

The epoch size has not been tested yet, and there is no approved experimental value for the epoch size to ensure that the datum won't exceed the size limitation. Testing or calculating the size is necessary to ensure that the system will not break.

Proposal

1. To optimize the voting process, it's crucial to determine appropriate settings for the quorum and voting duration. This can be achieved through research and experimentation based on the total amount of ADAO tokens in existence.

2. Modify the Peg Stability Module (PSM) proposal mechanism to enable the addition or removal of multiple stablecoins at once. Currently, the system only allows the addition or removal of one stablecoin per proposal.

3. Research and implement mechanisms to prevent any single ADAO holder from exerting disproportionate influence through 51% voting dominance.

Staking pool

It is recommended to incorporate leap year calculations into the reward calculation mechanism by utilizing data from an oracle. Leap year adjustments can enhance the accuracy and fairness of staking rewards distribution.