Paideia: Basic DAO contracts

These contracts cover the basic use case of a DAO, including on chain voting using governance tokens and spending from a shared treasury through these votes. In the first section we will dive into the general requirements we set for these contracts and in the following sections we describe a specification for an implementation supporting these requirements.

1 General requirements

These are requirements that are important for the solution to fulfill.

1. Creating a DAO should be permissionless.
2. DAO treasury should not depend on a datum to prevent funds being stuck and easy depositing.
3. It should not be possible to vote on the same proposal twice with the same governance tokens.
4. It should be possible to vote on two proposals running in parallel.
5. Proposal spam needs to be limited.
6. Every proposal must have a negative option to vote on, which results in no action taken

2 Solution overview

Bootstrapping a DAO is done by minting an NFT into a 'Paideia DAO' utxo with configurations defined in it's datum. Other validators used in a DAO are dependent on this NFT and utxo. Treasury is a spending validator which needs a valid 'DAO Action' being spent in the same transaction. A proposal is created by minting a 'DAO Proposal' token into a proposal utxo. This proposal utxo is used to keep track of vote tallies and when the voting ends, validation of initial values is done using the dao utxo as a reference input. The proposal will have 2 or more vote options defining the options a DAO member can vote on (with option 0 always being the negative option). Along with the proposal 0 or more action utxos can be created each referring to one of the non-zero options of the created proposal. A DAO member can vote on a proposal by initially locking governance tokens into a 'DAO Vote' utxo. Once the tokens are locked the member can only unlock their governance tokens if the Vote utxo does not contain any 'Vote receipt' tokens. A user votes on a proposal by spending their Vote utxo and creating a new one with 'Vote Receipt' tokens minted into it equal to the amount of voting power used. The 'Vote Receipt' token is unique to the proposal-option combination voted on. Vote Receipt tokens can be burned by either referencing a finished proposal or by removing the vote from an active proposal. The Vote utxo can have multiple different 'Vote Receipt' tokens in it, each belonging to a different proposal, but each receipt amount should not be larger than the amount of governance tokens.

3 Solution Elements

Descriptions of the elements that together form the Paideia DAO solution

3.1 Mint Policies & Assets

The assets used in the solutions with their mint and burn conditions and references to minting policies.

3.1.1 Minting Policy: Paideia DAO

This minting policy controls correct creation of new DAO's and helps identify each unique DAO.

3.1.1.1 Asset: DAO Identifier

A unique NFT that is minted into a newly created Paideia DAO utxo.

Minting Conditions	Burning Conditions
• Only 1 minted into a Paideia DAO utxo • Unique name • Valid datum in Paideia DAO utxo	• N/A

3.1.2 Minting Policy: Vote

The Vote mint policy deals with minting Vote NFT's for DAO members, ensuring their uniqueness.

3.1.2.1 Asset: Vote NFT

A unique NFT representing the locked governance tokens a DAO member can vote with, minted in combination with a reference NFT to provide CIP-68 metadata.

Minting Conditions	Burning Conditions
• Unique name • Governance tokens paid into Vote utxo	• Corresponding Vote utxo has no Vote Receipt assets remaining

3.1.3 Minting Policy: Proposal

A proposal minting policy will mint a unique Vote Receipt Asset for each voting option, giving simple tracking of voting.

3.1.3.1 Asset: Proposal Identifier

Minted into a proposal utxo, proving that the proposal is valid according to the configuration of the DAO.

Minting Conditions	Burning Conditions
• Proposal utxo it is minted in to is guarded by correct script • Proposal utxo datum is filled out correctly • Unique name	• N/A

3.1.3.2 Asset: Action Identifier

Minted into an action utxo, proving that the action is valid according to the configuration of the DAO.

Minting Conditions	Burning Conditions
• Action utxo it is minted in to is guarded by correct script • Action utxo datum is filled out correctly • Unique name	• Upon execution of the action OR • Upon proving the accompanied proposal has not passed

3.1.3.3 Asset: Vote Receipt

A unique asset for each proposal-option combination. When a DAO member votes on a proposal this will be minted into their Vote utxo.

Minting Conditions	Burning Conditions
• Minted into a Vote utxo • Amount smaller than or equal governance token amount in Vote utxo	• Corresponding proposal has ended and has been evaluated OR • User removes/changes their vote on this proposal

3.2 Spending validators

These are the spending validators that will guard the utxos that a Paideia DAO consists of.

3.2.1 Paideia DAO

One instance for each DAO will hold the configuration of it and control the creation of proposals and actions.

3.2.1.1 Assets

Policy	Name	Amount	Description
Paideia DAO	Unique	1	Unique identifier for this DAO

3.2.1.2 Datum

Field name	Type	Description
name	String	Name of the DAO
governance_token	ByteArray	Combination of policy id and asset name of governance token
threshold	Rational	Percentage of votes needed for a proposal option to pass
min_proposal_time	Int	Minimum amount of time a proposal's end time needs to be in the future
max_proposal_time	Int	Maximum amount of tine a proposal's end time is allowed to be in the future
quorum	Int	Amount of votes that need to be cast on a proposal to pass it
min_gov_proposal_create	Int	Amount of governance tokens must be present in Vote utxo to be allowed to create a Proposal
whitelisted_proposals	List<ByteArray>	Script hashes of supported proposal validators.
whitelisted_actions	List<ByteArray>	Script hashes of supported action validators.

3.2.2 Vote

A Vote utxo holds the governance tokens of a single DAO member along with Vote Receipts for proposals the member has voted on.

3.2.2.1 Assets

Policy	Name	Amount	Description
Vote	Unique reference NFT	1	Unique reference NFT that matches the DAO member's Vote NFT
-	Governance token	1-N	The governance tokens owned by the DAO member that can be used as voting power in this DAO
Proposal	Vote Receipt	0-N	Vote Receipts for proposals this DAO member has voted on

3.2.2.2 Datum

Field name	Type	Description
metadata	Dict	Metadata for the Vote NFT belonging to this Vote utxo

version	Int	CIP-68 version number
extra	None	No extra data needed

3.2.3 Proposal

Keeps track of total votes on an active proposal and ensures the proposal state is correctly set when the proposal end time has passed.

3.2.3.1 Assets

Policy	Name	Amount	Description
Proposal	Unique NFT	1	Unique NFT to identify this proposal

3.2.3.2 Datum

Field name	Type	Description
name	String	Name of the proposal
description	String	Description of the proposal
tally	List<Int>	Vote tally for the different vote options
end_time	Int	Time this proposal will end
status	ProposalStatus	The state this proposal is in

ProposalStatus

Type	Description
Active	Initial proposal status
FailedThreshold	Proposal failed due to not reaching threshold
FailedQuorum	Proposal failed due to not reaching quorum
Passed(Int)	Proposal option i has passed (Note: proposal option 0 always results in no action taken, so UI might show something less positive in case of Passed(0))

3.2.4 Action

An action is created at the same time as the proposal it refers to. It contains the specific settings for an action to be taken and ensures the action is taken correctly. In the current scope the only action type is sending funds from the treasury.

3.2.4.1 Assets

Policy	Name	Amount	Description
Proposal	Unique NFT	1	Unique NFT to identify this action

3.2.4.2 Datum

Field name	Type	Description
name	String	Name of the action
description	String	Description of the action
proposal	ByteArray	Reference to unique proposal NFT this action refers to
option	Int	The option that needs to pass for this action to be executed (can never be 0)

activation_time	Int	Time this action is activated (can be later than the end time of proposal)

3.2.5 Treasury

A simple validator that only allows spending if a valid action is part of the inputs. Any assets can be present and no datum requirements.

3.3 Transactions

The transactions are what transform the state of a DAO in a controlled way, using the rules set in the spending validators.

3.3.1 Create DAO

A user creates a DAO directly by spending into a Paideia DAO utxo with the desired configuration in it's datum and minting a unique identifier.

Inputs	Minting Policy	Outputs
• User	• Paideia DAO ‣ +1 DAO Identifier	• Paideia DAO ‣ DAO Identifier

3.3.2 Create Vote

A user becomes a DAO member by locking an amount of governance tokens into a Vote utxo and minting a Vote NFT to be used by the DAO member to control their vote.

Inputs	Minting Policy	Reference Input	Outputs
• User	• Vote ‣ +1 Reference Vote NFT ‣ +1 Vote NFT	• Paideia DAO ‣ DAO Identifier	• Vote ‣ Reference Vote NFT • User ‣ Vote NFT

3.3.3 Create Proposal

A DAO member creates a proposal and optionally its potential actions by proving membership with their Vote NFT and spending into Proposal and Action utxos.

Inputs	Minting Policy	Reference Input	Outputs
• User ‣ Vote NFT	• Proposal ‣ +1 Proposal Identifier ‣ +N Action Identifier	• Paideia DAO ‣ DAO Identifier • Vote ‣ Reference Vote NFT	• Proposal ‣ Proposal Identifier • Action ‣ Action Identifier • User ‣ Vote NFT

3.3.4 Cast Vote

A DAO member casts their vote by updating the proposal correctly and minting the correct amount of Vote Receipts into their Vote utxo.

Inputs	Minting Policy	Outputs
• Proposal ‣ Proposal Identifier • Vote	• Proposal ‣ +N Vote Receipt	• Proposal ‣ Proposal Identifier • Vote

‣ Reference Vote NFT ‣ N Governance Token • User ‣ Vote NFT		‣ Reference Vote NFT ‣ N Governance Token ‣ N Vote Receipt • User ‣ Vote NFT

3.3.5 Evaluate Proposal

Once the end time of the proposal has passed it's status can be set based on vote tally and configurations in the DAO.

Inputs	Reference Inputs	Outputs
• Proposal ‣ Proposal Identifier	• Paideia DAO ‣ DAO Identifier	• Proposal ‣ Proposal Identifier

3.3.6 Execute Action (Send Treasury funds)

If the proposal has passed with the option this action refers to it can be executed.

Inputs	Minting Policy	Reference Inputs	Outputs
• Treasury • Action ‣ Action Identifier	• Proposal ‣ −1 Action Identifier	• Paideia DAO ‣ DAO Identifier • Proposal ‣ Proposal Identifier	• Target

3.3.7 Clean Receipts

A DAO member can clean their Vote utxo by burning Vote Receipt tokens of proposals that have ended.

Inputs	Minting Policy	Reference Inputs	Outputs
• Vote ‣ Reference Vote NFT ‣ N Vote Receipt • User ‣ Vote NFT	• Proposal ‣ -N Vote Receipt	• Proposal ‣ Proposal Identifier	• Vote ‣ Reference Vote NFT • User ‣ Vote NFT

3.3.8 Empty Vote

A DAO member can (fully or partially) empty their Vote utxo of governance tokens as long as the remaining amount of Vote Receipt tokens does not exceed the remaining amount of governance tokens.

Inputs	Minting Policy	Outputs
• Vote ‣ Reference Vote NFT ‣ N Governance token • User ‣ Vote NFT	• Vote ‣ −1 Reference Vote NFT ‣ −1 Vote NFT	• User ‣ N Governance token

4 Implementation choices

During our work on Coinecta we have gained experience in using Aiken for on chain code and C# in combination with Pallas.NET for off chain code.

4.1 On chain

Besides Aiken and of course Plutus there are now numerous options for writing smart contracts, of which plu-ts is the most similar to Aiken. Due to our previous experience with Aiken we feel that using Aiken will give us a fair boost in productivity and no clear con's. As we have no Haskell experience we feel using Plutus will give us a large learning curve for little benefit over Aiken. So our on chain language of choice will be Aiken.

4.2 Off chain

The off chain code could be made in many different languages, but both for productivity's sake and for improving open source availabiltity we will be using C# in the backend due to our experience with the language and the relative small footprint in Cardano's open source ecosystem.

4.3 Tech List

- Aiken
  - sundae/multisig
- C#
  - Pallas.NET
  - Cborserializer
  - SAIB.CardanoSharp.Wallet