TX META

Decentralized Credit Unions Toolkit

Design Specification

Contents

Decentralized Credit Unions (DCU) Toolkit

1. Overview

The DCU-Toolkit (Decentralized Credit Unions Toolkit) is a smart contract infrastructure developed using Aiken for the Cardano blockchain. It is designed to facilitate automated cooperative finance operations including group savings, rotating fund distribution, democratic governance, and treasury management for traditional savings groups such as Chamas, SACCOs, Tontines, and similar cooperative finance models.

This toolkit empowers members to seamlessly create accounts, form cooperative groups, contribute funds, participate in democratic decision-making, and manage shared treasuries directly from their wallets. It ensures secure and efficient transactions by automating group governance, fund rotation, and treasury operations within a decentralized framework.

2. Architecture

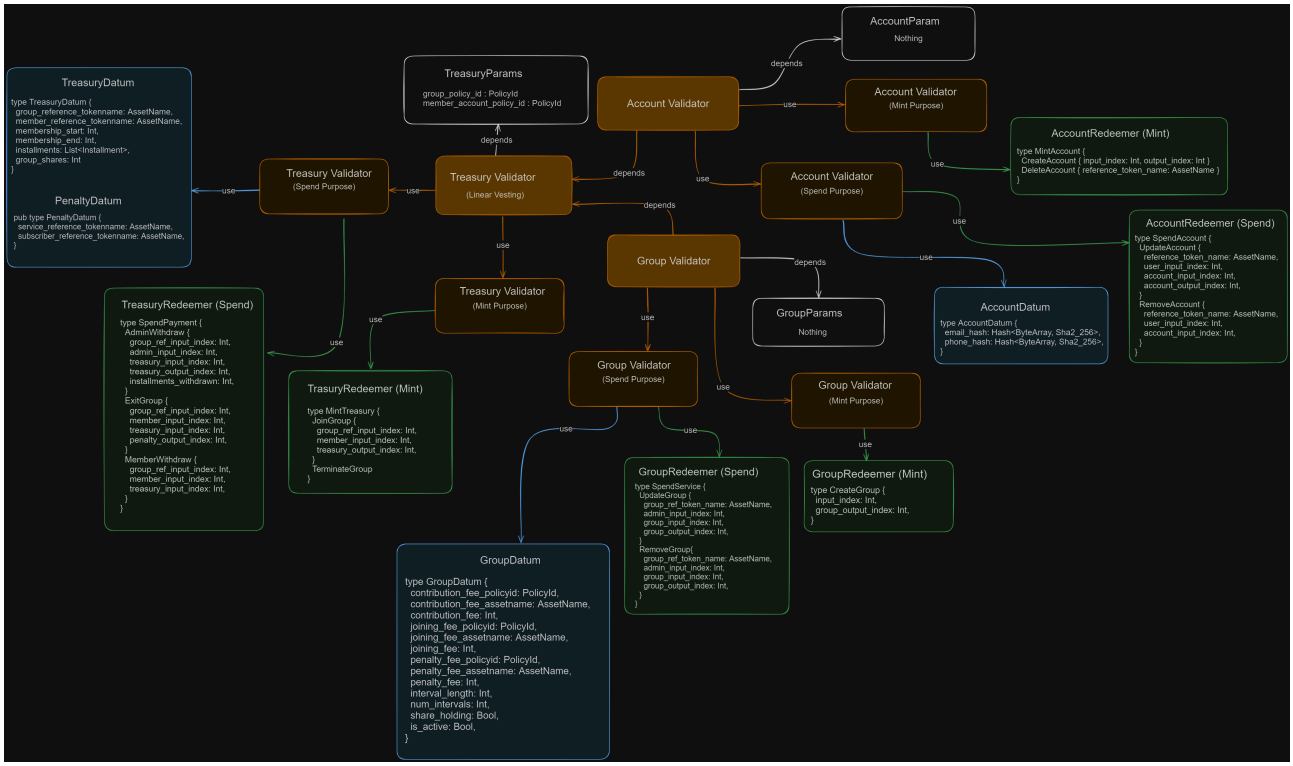


Figure 1: DCU Toolkit Architecture

There are three validators in this cooperative finance system.

1. **Account Validator**

   A multi-validator responsible for creating member accounts by minting CIP-68 compliant Account NFT Assets and sending the user NFT to the member's wallet while sending the reference NFT to the spending endpoint. It enables members to update their account metadata and delete their accounts by burning the Account NFTs.

2. **Group Validator**

   A multi-validator responsible for creating cooperative groups by minting CIP-68 compliant Group NFT Assets. It manages group configuration including contribution fees, joining fees, penalties, subscription intervals, and democratic governance rules. The validator enables group administrators to update group metadata and deactivate groups when necessary given a signature threshold.

3. **Treasury Validator**

This is the core validator responsible for managing prepaid contributions, rotating fund distribution, member participation, and withdrawal operations. The contract incorporates a linear vesting mechanism to gradually release funds to members according to the group's rotation schedule. It handles member joins, exits, administrative withdrawals, and penalty management within the cooperative finance framework.

3. Specification

3.1. System Actors

1. **Member**

   An entity who interacts with the Account Validator to create an account and join cooperative groups by depositing contributions to the Treasury Validator. A user becomes a member when they mint an Account NFT and can participate in multiple cooperative groups.

2. **Group Administrator**

   An entity who interacts with the Group Validator to create cooperative groups and manage group configurations. A member becomes a group administrator when they create a group by minting a Group NFT. Administrators can update group parameters, manage member participation, and withdraw funds according to group rules.

3.2. Tokens

1. **Account NFT**

   Can only be minted by a user when creating an account in the cooperative finance system and burned when the user deletes their account. A check must be included to verify that there are no active memberships in any Treasury before burning.

   - **TokenName:** Defined in Account Validator using CIP-68 standards with transaction ID, output index, and appropriate prefix (`prefix_100` for reference NFT, `prefix_222` for user NFT).

2. **Group NFT**

   Can only be minted when creating a cooperative group and held by the group administrator. This NFT represents ownership and administrative control over the group configuration and operations.

   - **TokenName:** Defined in Group Validator using CIP-68 standards with transaction ID, output index, and appropriate prefix (`prefix_100` for reference NFT, `prefix_222` for user NFT).

3. **Membership NFT**

   Can only be minted when a user joins a group and the Reference NFT is locked in the Treasury Validator while the User NFT is sent to the member.

   - **TokenName:** Defined in Treasury Validator using CIP-68 standards with transaction ID, output index, and appropriate prefix (`prefix_100` for reference NFT, `prefix_222` for user NFT).

3.3. Smart Contracts

3.3.1. Treasury Multi-validator

The Treasury Validator is the core contract responsible for managing member contributions, validating group memberships, and ensuring proper distribution of funds through rotating schedules and linear vesting. It facilitates member joins, exits, fund withdrawals, and penalty processing, allowing both members and administrators to interact securely within the cooperative finance framework.

3.3.1.1. Parameters

- **`group_policy_id`** : Hash of the Group PolicyId
- **`member_account_policy_id`** : Hash of the Account PolicyId

3.3.1.2. Minting Purpose

3.3.1.2.1. Redeemer

- ```
  JoinGroup {
    group_ref_input_index: Int,
    member_input_index: Int,
    treasury_output_index: Int,
  }
  ```
- **`TerminateGroup`**

3.3.1.2.2. Validation

1. **JoinGroup**

   The redeemer allows a member to join a cooperative group by minting one unique Membership Token representing their membership.

   - Validate that the member's Account NFT is present in the transaction inputs.
   - A Group Input must be provided from the Group Validator (as a spending input, not reference) to update the member count.
  ```

- The treasury output must be sent to the Treasury Script's address and contain a Treasury datum that is consistent with the Group datum.
- Exactly one Membership Token (with token name "treasury-membership") is minted.
- Validate that sufficient contribution fees and joining fees are locked in the Treasury UTxO according to the Group datum specifications.
- Ensure that the User NFT doesn't go to the Script
- Ensure Membership token goes back to the script

2. **TerminateGroup**
   - The redeemer must burn exactly one Membership Token (i.e. a single token with the token name "treasury-membership" is burned).
   - Validate that the Group reference input provides valid Group datum.

3.3.1.3. Spend Purpose

3.3.1.4. Datum

This is a Sum type datum where one represents the treasury datum and the other represents a penalty datum.

3.3.1.4.1. Treasury datum

- **`group_reference_tokenname: AssetName`** – Links to the Group Validator.
- **`member_reference_tokenname: AssetName`** – Identifies the member's Account NFT.
- **`membership_start: Int`** – The timestamp when the member joined the group.
- **`contribution_list:`** List of contributions – Each contribution specifies when and how much can be withdrawn based on the rotation schedule.
  - **`Contribution`**:
    - **`claimable_at : Int`** – Time after which the loan can be claimed.
    - **`claimable_amount : Int`** – The loan amount available for withdrawal at that time.

3.3.1.4.2. Penalty datum

- **`group_reference_tokenname: AssetName`** – Links to the Group Validator.
- **`member_reference_tokenname: AssetName`** – Identifies the member's Account NFT.

3.3.1.5. Redeemer

- ```
  DistributePayout {
    group_ref_input_index: Int,
    treasury_input_indices: List<Int>,
    treasury_output_indices: List<Int>,
    borrower_output_index: Int,
  }
  ```
- ```
  ExitGroup {
    group_ref_input_index: Int,
    member_input_index: Int,
    treasury_input_index: Int,
    penalty_output_index: Int,
  }
  ```
- ```
  MemberWithdraw {
    group_ref_input_index: Int,
    member_input_index: Int,
    treasury_input_index: Int,
    treasury_output_index: Int,
    loans_withdrawn: Int,
  }
  ```

3.3.1.6. Validation

1. **DistributePayout**

   This redeemer allows the group (or an administrator/bot) to trigger the distribution of funds to the eligible member for the current interval. It utilizes a "trustless collection" pattern where multiple member inputs are aggregated and sent directly to the borrower.

   - **Eligibility Check**: Validate that the "Borrower" (recipient at **`borrower_output_index`**) provides a **Slot NFT** that matches the designated slot for the current interval (derived from the **`group_datum`** and time). This strictly enforces the rotation schedule.
   - The `Treasury` UTxOs being spent (at **`treasury_input_indices`**) must be identified and contain valid Treasury datums.
  ```

- A Group datum must be supplied as a reference input (at **`group_ref_input_index`**) to determine the Rotation Schedule and current Borrower.
- The "Borrower" output (at **`borrower_output_index`**) must go to the address of the scheduled member.
- The value sent to the borrower must equal the sum of the collected `claimable_amounts` from the inputs (minus any transaction fees if applicable).
- The outputs at **`treasury_output_indices`** must return the Treasury UTxOs to the script address with updated datums (marking the current interval as "paid").

2. **ExitGroup**

   The redeemer allows a member to exit a cooperative group, unlocking remaining contributions to their wallet address provided they have no loans.

   - The member must provide an input (at **`member_input_index`**) containing the appropriate Account NFT.
   - The Treasury UTxO (from treasury_input_index) must have a valid Treasury datum.
   - The Group UTxO is provided as a spending input to decrement the member count.
   - The decision branch is based on the membership timing:
     - **Penalty:** If exiting early (active group), the transaction must produce an output (at penalty_output_index) carrying a Penalty datum. This output must include at least the minimum penalty fee as defined by the Group datum.

3. **MemberWithdraw**

   The redeemer allows a member to withdraw their allocated funds when it's their turn in the rotation schedule.

   - The member's input (at member_input_index) must contain the correct Account NFT.
   - The Treasury UTxO (from treasury_input_index) must have a valid Treasury datum.
   - The Group datum (from the reference input at group_ref_input_index) must be validated for withdrawal eligibility.
   - Implement linear vesting by verifying withdrawal amount against vesting schedule and member's share allocation, contribution and reputation score.
   - If group is inactive, allow full withdrawal and burn the Membership Token.

3.3.2. Group Validator

The Group Validator is responsible for creating cooperative groups, managing group configurations, and controlling group lifecycle operations including activation and deactivation.

3.3.2.1. Parameter

Nothing

3.3.2.2. Minting Purpose

3.3.2.2.1. Redeemer

- CreateGroup

3.3.2.2.2. Validation

1. **CreateGroup**

   The redeemer allows creating a new cooperative group by minting one unique CIP-68 compliant Group Token.

   - An input (at `input_index`) must be present to derive unique token names (using CIP68 prefixes).
   - Validate that exactly one Reference Token and one User Token are minted as per the CIP68 compliance standards.
   - The unique tokens are derived from the transaction ID and output index of the input.
   - The output at `group_output_index` must be sent to the Group Script's address.
   - The output must contain a Group datum with the following requirements:
     - **contribution_fee**: Must be greater than 0.
     - **joining_fee**: Must be ≥ 0.
     - **penalty_fee**: Must be ≥ 0.
     - **interval_length**: Must be greater than 0.
     - **num_intervals**: Must be > 0 and within a reasonable bound (e.g. ≤ 100).
     - **member_count**: Must be initialized to 0.
     - **is_active**: Must be set to true.

3.3.2.3. Spend Purpose

3.3.2.3.1. Datum

- **`contribution_fee_policyid: PolicyId`** The PolicyId governing the asset used for the contribution fee.
- **`contribution_fee_assetname: AssetName`** The AssetName of the contribution fee.
- **`contribution_fee: Int`** An Int representing the contribution fee amount per interval.
- **`joining_fee_policyid: PolicyId`** The PolicyId governing the asset used for the joining fee.
- **`joining_fee_assetname: AssetName`** The AssetName of the joining fee.
- **`joining_fee: Int`** An Int representing the one-time joining fee.
- **`penalty_fee_policyid: PolicyId`** The PolicyId governing the asset used for the penalty fee.
- **`penalty_fee_assetname: AssetName`** The AssetName of the penalty fee.
- **`penalty_fee: Int`** An Int representing the fee deducted when a member exits early.
- **`interval_length: Int`** An Int defining the duration of one contribution interval.
- **`num_intervals: Int`** An Int representing the total number of intervals in the rotation cycle.
- **`member_count: Int`** An Int tracking the number of active members in the group.
- **`share_holding: Bool`** A Bool indicating if members can hold multiple shares.
- **`is_active: Bool`** A Bool indicating whether the group is currently active.

Note: Contribution fees can be based on length of period the member commits to, e.g. If they pay for one cycle, the fees may differ from paying for multiple cycles. Group administrators can configure flexible pricing models.

3.3.2.3.2. Redeemer

- ```
  UpdateGroup {
    group_ref_token_name: AssetName,
    admin_input_index: Int,
    group_input_index: Int,
    group_output_index: Int,
  }
  ```
  *

  - *```rust
    RemoveGroup {
      group_ref_token_name: AssetName,
  ```

```
  admin_input_index: Int,
  group_input_index: Int,
  group_output_index: Int,
}
```

3.3.2.3.2.1. Validation

1. **UpdateGroup**

   This redeemer endpoint allows the administrator to update the group metadata attached to a Group UTxO.

   - A Group UTxO containing the Group NFT must be provided (at group_input_index) with its output reference matching the provided reference.
   - An administrator input (at admin_input_index) must be present to prove ownership of the Group NFT (derived from group_ref_token_name).
   - The output at group_output_index must be sent to the Group Script's address and must include an updated Group datum.
   - Validate that the metadata of the Reference NFT token is updated within acceptable bounds.
   - Critical metadata changes (fees, intervals) are **ONLY** allowed if `member_count == 0`.
   - The Group NFT must still be present in the output.

2. **RemoveGroup**

   This redeemer endpoint allows an administrator to deactivate a group from the cooperative finance system.

   - The transaction must include two script inputs:
     - One input containing the Group UTxO with the Group NFT (at **`group_input_index`**).
     - An administrator input (at **`admin_input_index`**) proving ownership of the Group NFT.
   - Two script outputs must be produced, with one of them (at **`group_output_index`**) sent to the Group Script's address.
   - The output Group datum must indicate that the group is inactivated by setting is_active to false.
   - The Group NFT must still be present in the output to maintain correct state tracking.
   - `member_count` must be 0 to remove/deactivate a group.

3.3.3. Account Validator

The Account Validator handles the creation, update, and removal of member accounts within the cooperative finance system.

3.3.3.1. Parameter

Nothing

3.3.3.2. Minting Purpose

3.3.3.2.1. Redeemer

- `CreateAccount { input_index: Int, output_index: Int }`
- `DeleteAccount { reference_token_name: AssetName }`

3.3.3.2.1.1. Validation

1. **CreateAccount**

   The redeemer allows creating a new member account by minting one unique CIP-68 compliant Account Token.

   - An input must be present to derive unique token names using CIP68 prefixes.
   - Validate that exactly one Account Reference Token and one Account User Token are minted and the unique tokens are generated from the transaction's ID and output index.
   - Ensure the output (at `output_index`) must be sent to the Account Script's address and must carry an Account datum.
   - Ensure the datum includes valid account detail:
     - `email_hash`: Must be 32 bytes long, or
     - `phone_hash`: Must be 32 bytes long.
   - The User NFT must not be sent to the script.
   - The Reference NFT must be preserved at the script address.

2. **DeleteAccount**

This redeemer endpoint allows for the removal of a member account by burning the associated Account Tokens.

A Check That there are no active memberships should be done off-chain.

- Validate that the redeemer only burns one Account Reference Token and one Account User Token.
- There should be no remaining account-related tokens in the transaction after burning.

3.3.3.3. Spend Purpose

3.3.3.3.1. Datum

- **`email_hash: Hash<ByteArray, Sha2_256>:`** A hash (using Sha2_256) of the member's email as a ByteArray. This must be exactly 32 bytes long.
- **`phone_hash: Hash<ByteArray, Sha2_256>:`** A hash (using Sha2_256) of the member's phone number as a ByteArray. This must also be exactly 32 bytes long.

3.3.3.3.2. Redeemer

- ```
  UpdateAccount {
    reference_token_name: AssetName,
    user_input_index: Int,
    account_input_index: Int,
    account_output_index: Int,
  }
  ```
- ```
  RemoveAccount {
    reference_token_name: AssetName,
    user_input_index: Int,
    account_input_index: Int,
  }
  ```

3.3.3.3.2.1. Validation

1. **UpdateAccount**

   This redeemer endpoint allows a member to update the metadata attached to an Account UTxO.

   - Validate that an Account UTxO containing the Account NFT must be present in the inputs (at **`account_input_index`**).
   - A user input (at **`user_input_index`**) must include the Account User Token, proving ownership.

- The output (at **`account_output_index`**) must be sent to the Account Script's address and it must carry an updated Account datum.
- The updated Account datum must satisfy metadata validation, ensuring that contact details remain correctly formatted.
- The Reference NFT must be forwarded correctly to the spending endpoint.

2. **RemoveAccount**

   The redeemer allows the removal of an account by a member from the cooperative finance system.

   Must Check That there are no active memberships in the off-chain code.

   - The transaction must include an Account UTxO (at **`account_input_index`**) containing the Account NFT.
   - A user input (at **`user_input_index`**) must be present to prove ownership via the Account User Token.
   - The redeemer must burn the Account Reference NFT, which is validated by confirming that the minted value includes a burn (i.e. a negative quantity) for the reference token.

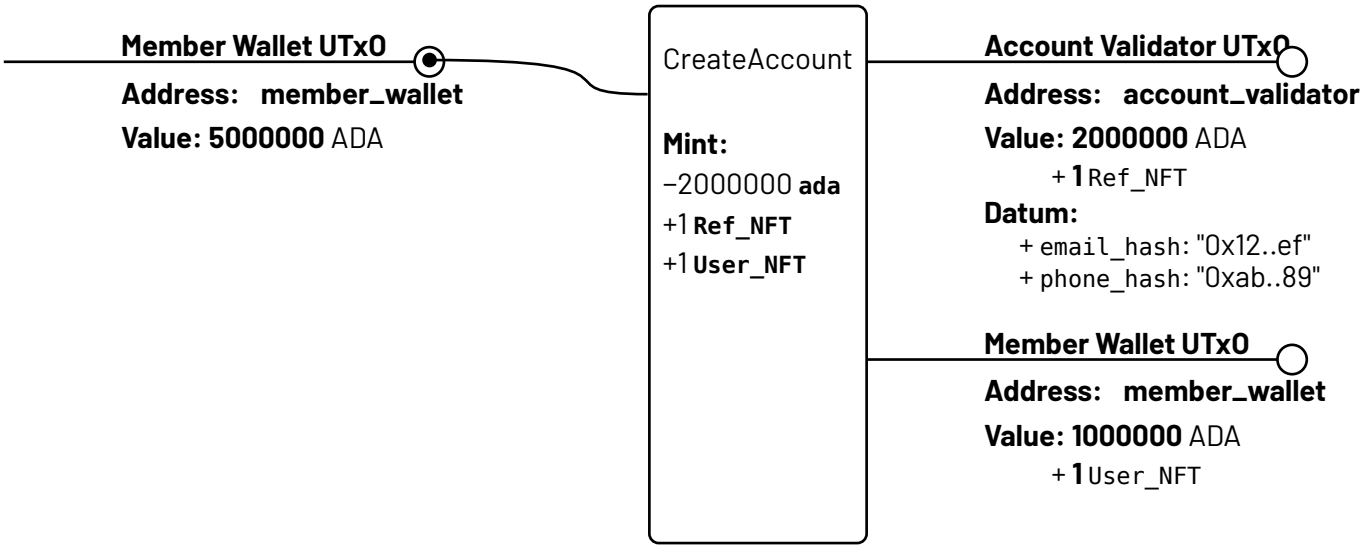
4. Transactions

This section outlines the various transactions involved in the DCU-Toolkit on the Cardano blockchain. Each transaction type demonstrates the interaction patterns between members, administrators, and the three core validators.

4.1. Account Validator

4.1.1. Mint :: CreateAccount

This transaction creates a new member account by minting Account NFTs. This transaction is performed by a user to establish their identity within the cooperative finance system and enable participation in multiple groups.



Note: Create Account Transaction

4.1.1.1. Inputs

1. **Member Wallet UTxO.**
   - Address: Member's wallet address
   - Value:
     - Minimum ADA
     - Any ADA required for the transaction.

4.1.1.2. Mints

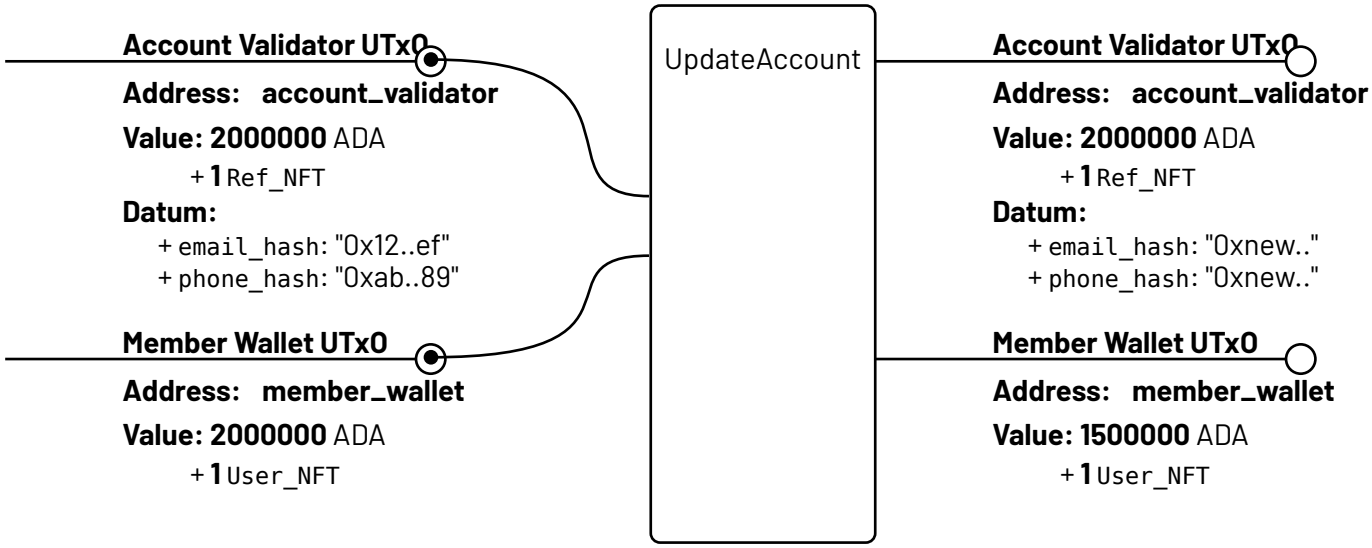
1. **Account Multi-validator**
   - Redeemer: CreateAccount
   - Value:
     - +1 Account NFT Asset
     - +1 Reference NFT Asset

4.1.1.3. Outputs

1. **Member Wallet UTxO:**
   - Address: Member wallet address
     - minimum ADA
     - 1 Account NFT Asset
2. **Account Validator UTxO:**
   - Address: Account Validator script address
   - Datum:
     - **`email_hash: Hash<ByteArray, Sha2_256>:`** A hash (using Sha2_256) of the member's email as a ByteArray. This must be exactly 32 bytes long.
     - **`phone_hash: Hash<ByteArray, Sha2_256>:`** A hash (using Sha2_256) of the member's phone number as a ByteArray. This must also be exactly 32 bytes long.
   - Value:
     - 1 Account Reference NFT Asset

4.1.2. Spend :: UpdateAccount

This transaction updates the member's account metadata. It consumes both the Account NFT and the Reference NFT, then sends the updated Account NFT back to the member's wallet and the updated Reference NFT to the spending endpoint.



Note: Update Account Metadata

4.1.2.1. Inputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - Minimum ADA
     - 1 Account NFT Asset
2. **Account Validator UTxO**
   - Address: Account validator script address
   - Datum:
     - existing_metadata: listed in Section 3.3.3.3.1

.

- Value:
  - Minimum ADA

- 1 Reference NFT Asset

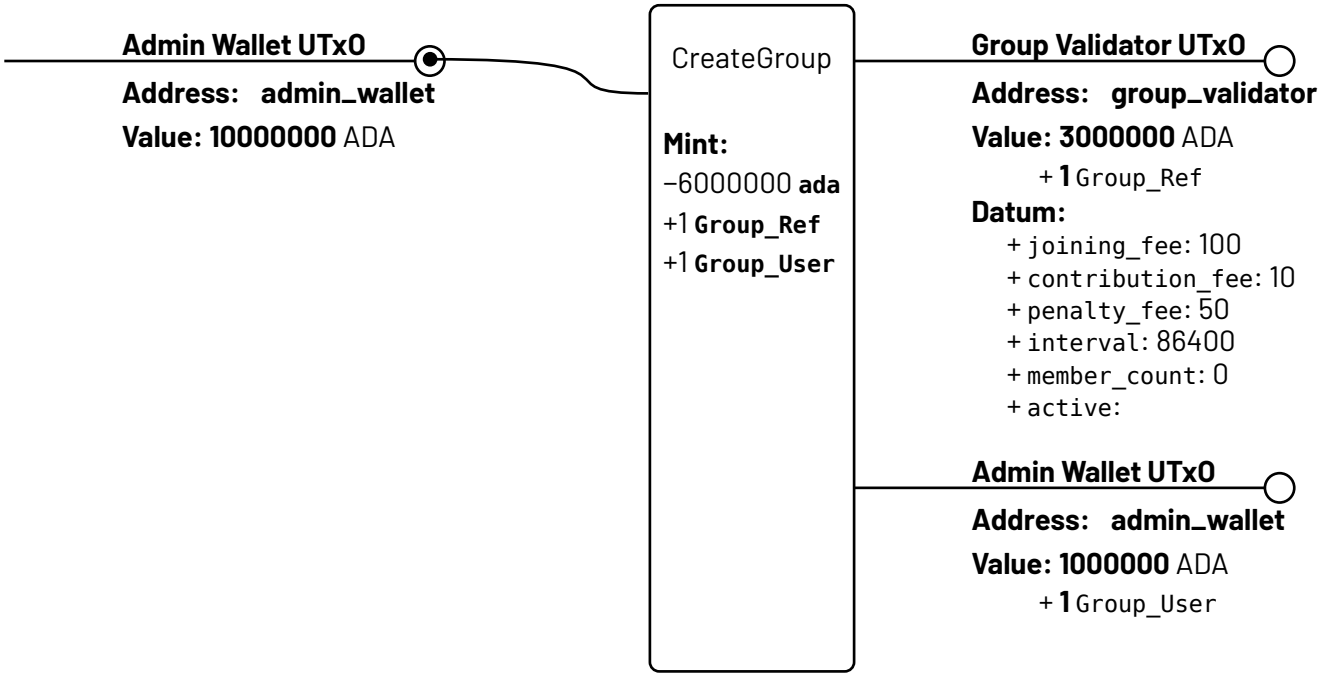
4.1.2.2. Outputs

1. **Member Wallet UTxO**
   - Address: Member wallet address
   - Datum:
     - updated_metadata: New metadata for the account.
   - Value:
     - Minimum ADA
     - 1 Account NFT Asset
2. **Account Validator UTxO:**
   - Address: Account validator script address
   - Datum:
     - updated_metadata: New metadata for the account
   - Value:
     - Minimum ADA
     - 1 Reference NFT Asset

4.2. Group Validator

4.2.1. Mint :: CreateGroup

This transaction creates a new cooperative group by minting Group NFTs. This transaction is performed by a member who wishes to become a group administrator.



Note: Create Group Transaction

4.2.1.1. Inputs

1. **Administrator Wallet UTxO.**
   - Address: Administrator's wallet address
   - Value:
     - Minimum ADA
     - Any additional ADA required for the transaction

4.2.1.2. Mints

1. **Group Multi-validator**

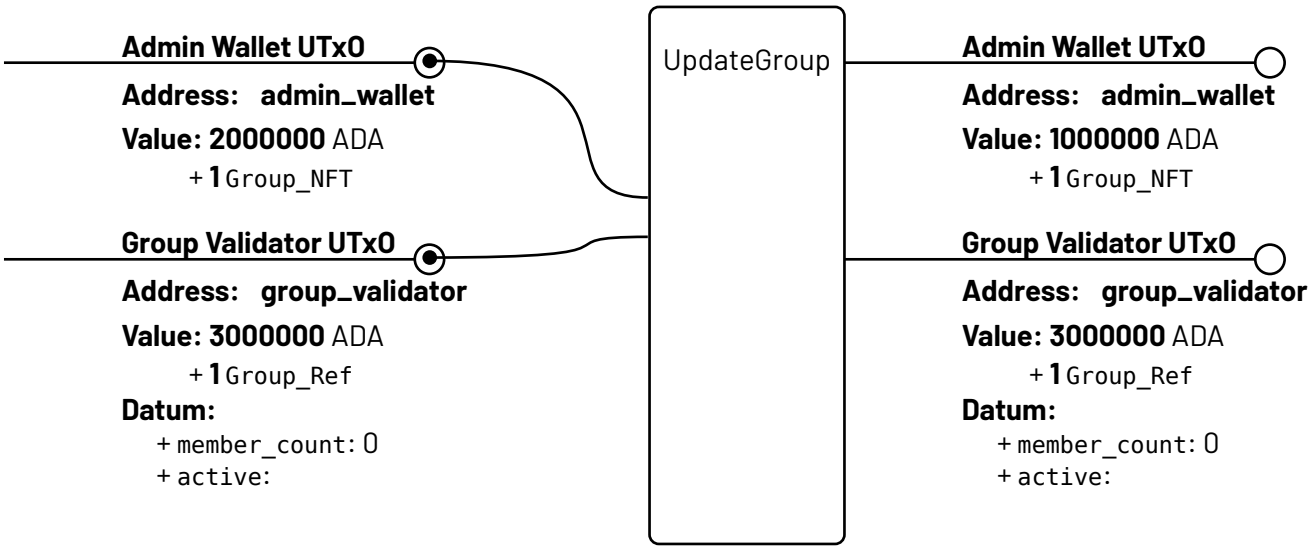
- Redeemer: CreateGroup
- Value:
  - +1 Group NFT Asset
  - +1 Reference NFT Asset

4.2.1.3. Outputs

1. **Administrator Wallet UTxO:**
   - Address: Administrator wallet address
   - Value:
     - minimum ADA
     - 1 Group NFT Asset
2. **Group Validator UTxO:**
   - Address: Group validator script address
   - Datum:
     - **`contribution_fee_policyid: PolicyId`** The PolicyId governing the asset used for the contribution fee.
     - **`contribution_fee_assetname: AssetName`** The AssetName of the contribution fee.
     - **`contribution_fee: Int`** An Int representing the contribution fee amount per interval.
     - **`joining_fee_policyid: PolicyId`** The PolicyId governing the asset used for the joining fee.
     - **`joining_fee_assetname: AssetName`** The AssetName of the joining fee.
     - **`joining_fee: Int`** An Int representing the one-time joining fee.
     - **`penalty_fee_policyid: PolicyId`** The PolicyId governing the asset used for the penalty fee.
     - **`penalty_fee_assetname: AssetName`** The AssetName of the penalty fee.
     - **`penalty_fee: Int`** An Int representing the fee deducted when a member exits early.
     - **`interval_length: Int`** An Int defining the duration of one contribution interval.
     - **`num_intervals: Int`** An Int representing the total number of intervals in the rotation cycle.
     - **`share_holding: Bool`** A Bool indicating if members can hold multiple shares.
     - **`is_active: Bool`** A Bool indicating whether the group is currently active.
   - Value:
     - 1 Group Reference NFT Asset

4.2.2. Spend :: UpdateGroup

This transaction updates the group metadata attached to the Group UTxO at the script address. It enables administrators to adjust group parameters such as fees, intervals, and governance rules.



Note: Update Group Metadata (Safe)

4.2.2.1. Inputs

1. **Administrator Wallet UTxO**
   - Address: Administrator's wallet address
   - Value:
     - Minimum ADA
     - Group NFT Asset
2. **Group Validator UTxO**
   - Address: Group validator script address
   - Datum:
     - existing_metadata: Current metadata listed in Section 3.3.2.3.1.
   - Value:
     - Minimum ADA
     - 1 Reference NFT Asset

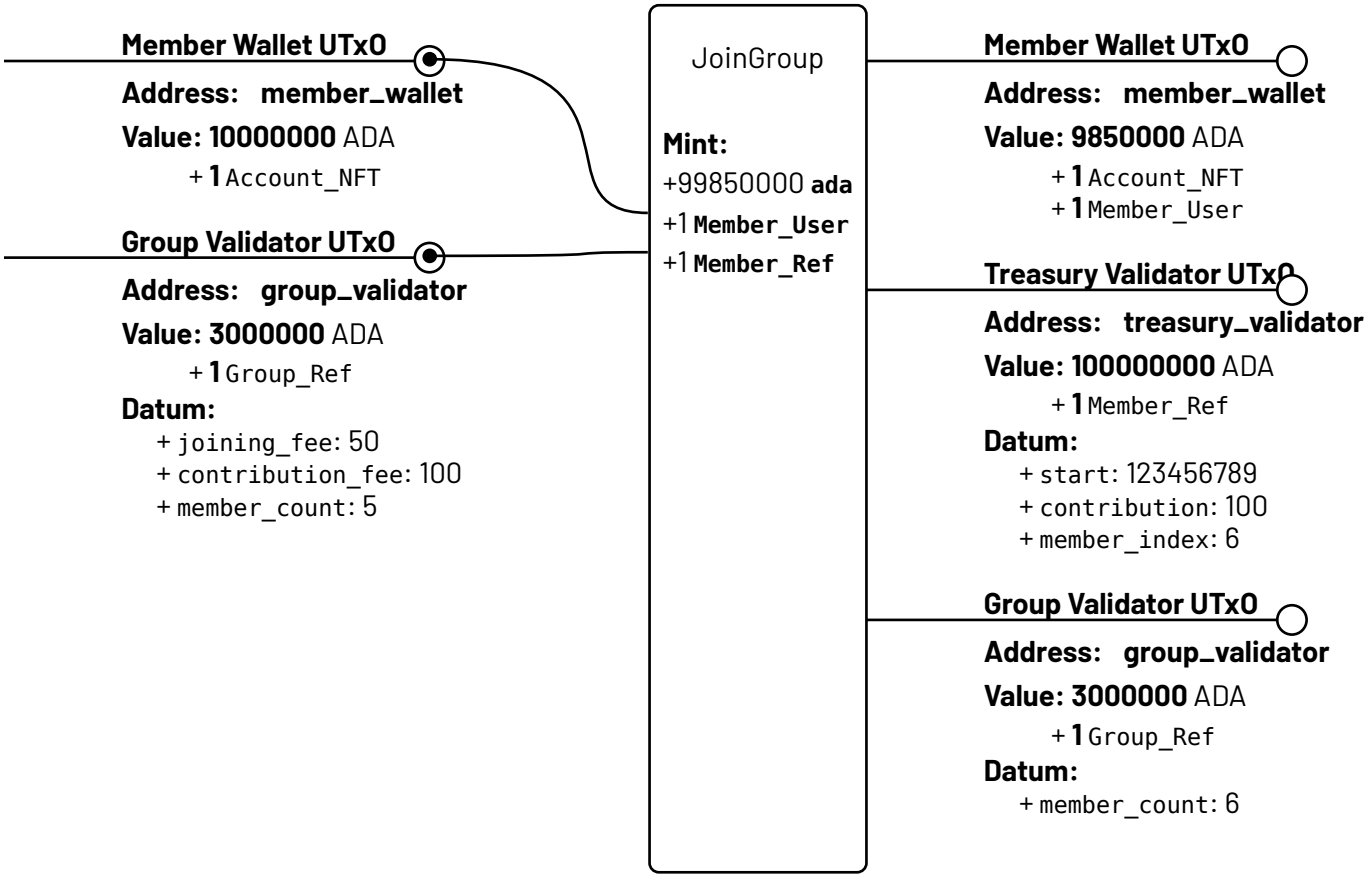
4.2.2.2. Outputs

1. **Administrator Wallet UTxO**
   - Address: Administrator's wallet address
   - Value:
     - Minimum ADA
     - 1 Group NFT Asset
2. **Group Validator UTxO**
   - Address: Group validator script address
   - Datum:
     - updated_metadata: updated metadata for the group listed in Section 3.3.2.3.1
   - Value:
     - Minimum ADA
     - 1 Reference NFT Asset

4.3. Treasury Validator

4.3.1. Mint :: JoinGroup

This transaction occurs when a member joins a cooperative group by locking contribution funds in the Treasury Validator script address.



Note: Join, Mint Slot Key & Lock Funds

4.3.1.1. Inputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - 100 ADA: Contribution amount to lock in the Treasury Contract.
     - 1 Account NFT Asset

2. **Group Reference UTxO**
   - Address: Group Contract Address
   - Datum:
     - group_datum: listed in Section 3.3.2.3.1
   - Value:
     - 1 Group Reference NFT Asset
     - Minimum Ada

4.3.1.2. Mints

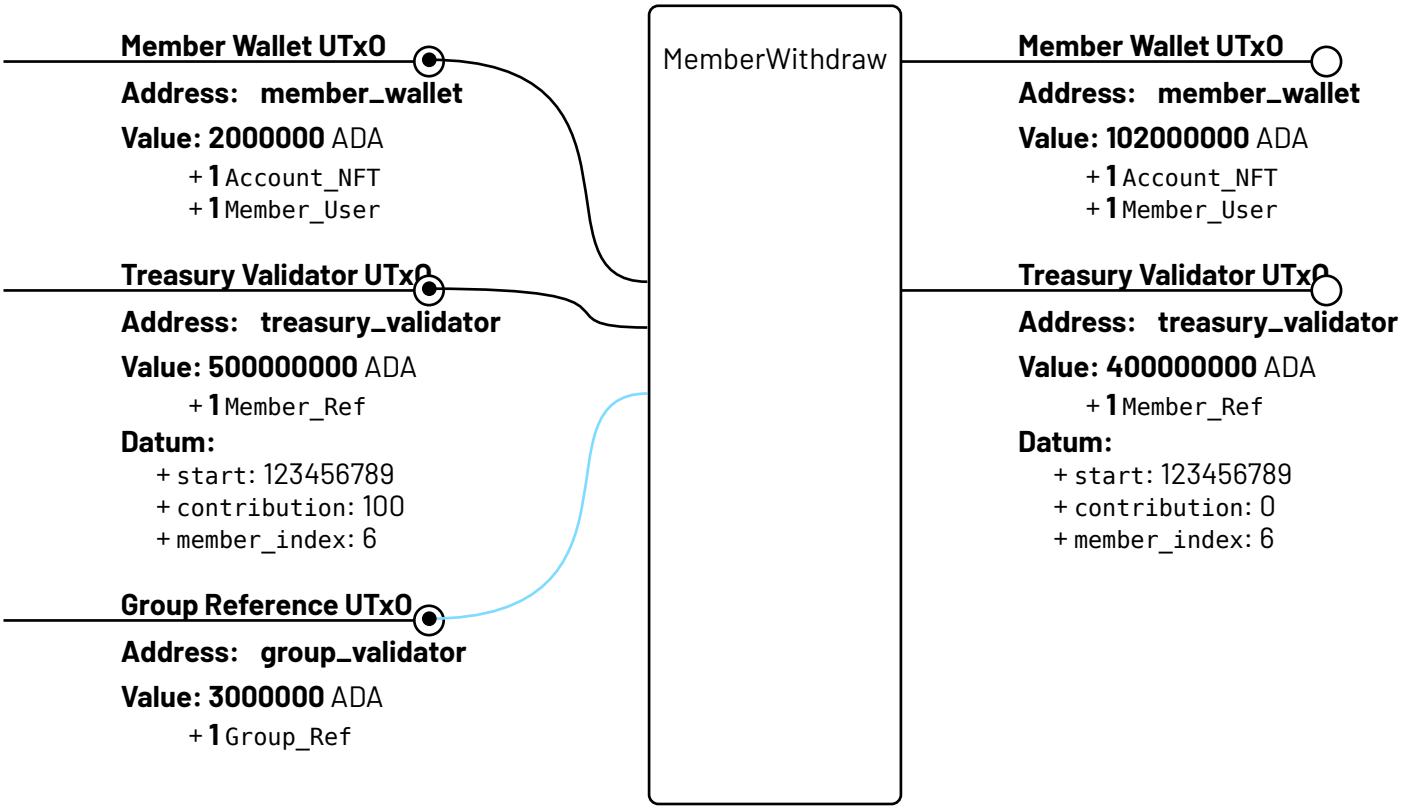
1. **Treasury Validator**
   - Redeemer: JoinGroup
   - Value:
     - +1 Treasury NFT Asset

4.3.1.3. Outputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - Change ADA
     - 1 Account NFT Asset
2. **Treasury Validator UTxO**
   - Address: Treasury validator script address
   - Datum:
     - treasury datum as listed in Section 3.3.1.4.1
   - Value:
     - 100 ADA: Contribution funds to be managed by the group
     - 1 Treasury NFT Asset

4.3.2. Spend :: MemberWithdraw

This transaction allows a member to withdraw their allocated funds when it's their turn in the rotation schedule.



Note: Member Withdraws Funds

4.3.2.1. Inputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - Minimum ADA
     - 1 Account NFT Asset
2. **Treasury Validator UTxO**
   - Address: Treasury validator script address
   - Datum:
     - datum listed in Section 3.3.1.4.1

- Value:
  - Locked contribution funds
  - 1 Treasury NFT Asset

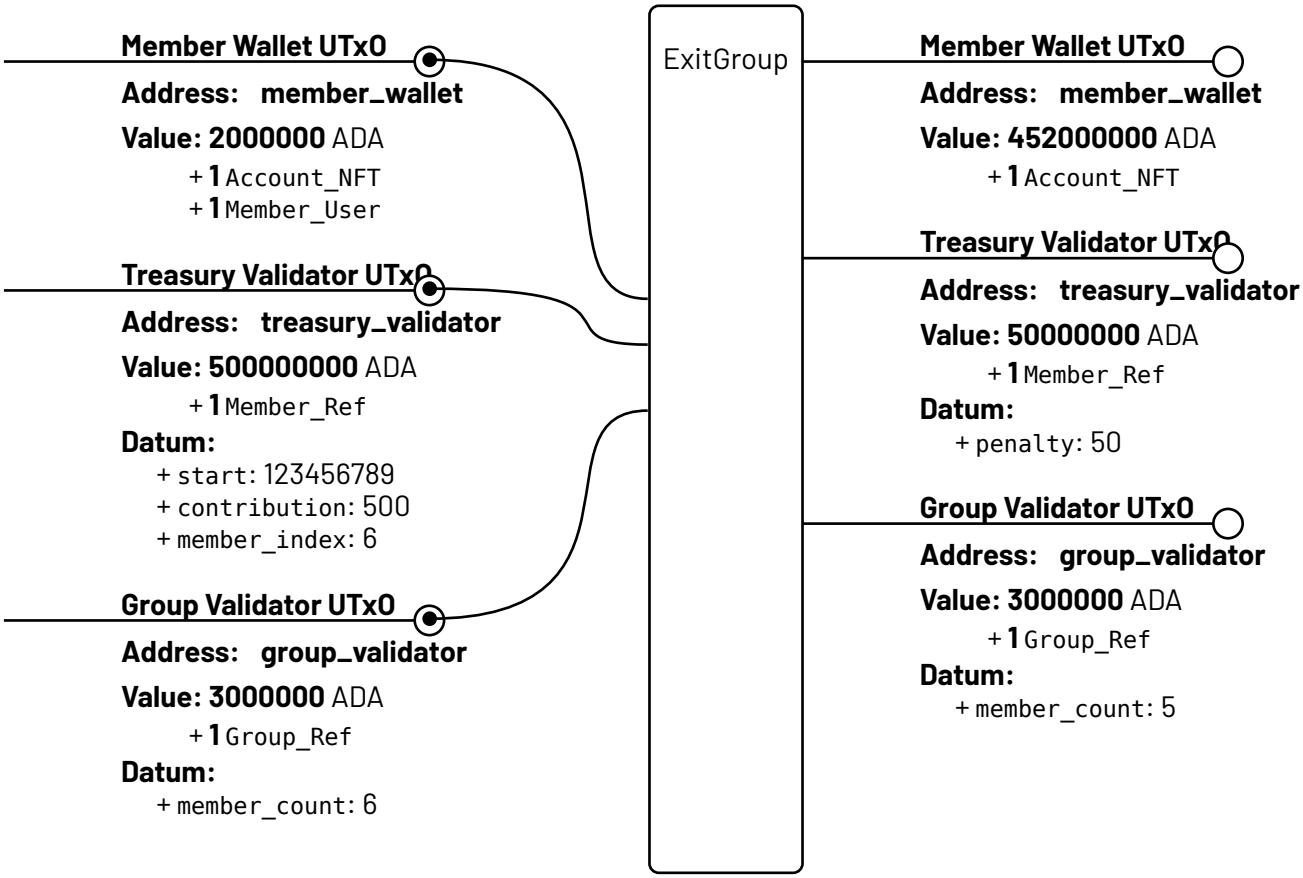
3. **Group Reference UTxO**
   - Address: Group Contract Address
   - Datum:
     - group_datum: listed in Section 3.3.2.3.1
   - Value:
     - 1 Group Reference NFT Asset
     - Minimum Ada

4.3.2.2. Outputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - Minimum ADA
     - Withdrawn funds according to vesting schedule
     - 1 Account NFT Asset
2. **Treasury Validator UTxO**
   - Address: Treasury validator script address
   - Datum:
     - datum listed in Section 3.3.1.4.1 with updated installments
   - Value:
     - Remaining funds after withdrawal
     - 1 Treasury NFT Asset

4.3.3. Spend :: ExitGroup

This transaction allows a member to exit a cooperative group by spending a Treasury UTxO, unlocking the remaining contribution to their wallet address and potentially creating a Penalty UTxO.



Note: Exit Group & Update Count

4.3.3.1. Inputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - Minimum ADA
     - 1 Account NFT Asset
2. **Treasury Validator UTxO**
   - Address: Treasury validator script address

- Datum:
  - current_datum: Current treasury metadata listed in Section 3.3.1.4.1
- Value:
  - Remaining contribution funds
  - Treasury NFT Asset

3. **Group Reference UTxO**
   - Address: Group Contract Address
   - Datum:
     - group_datum: listed in Section 3.3.2.3.1
   - Value:
     - 1 Group Reference NFT Asset
     - Minimum Ada

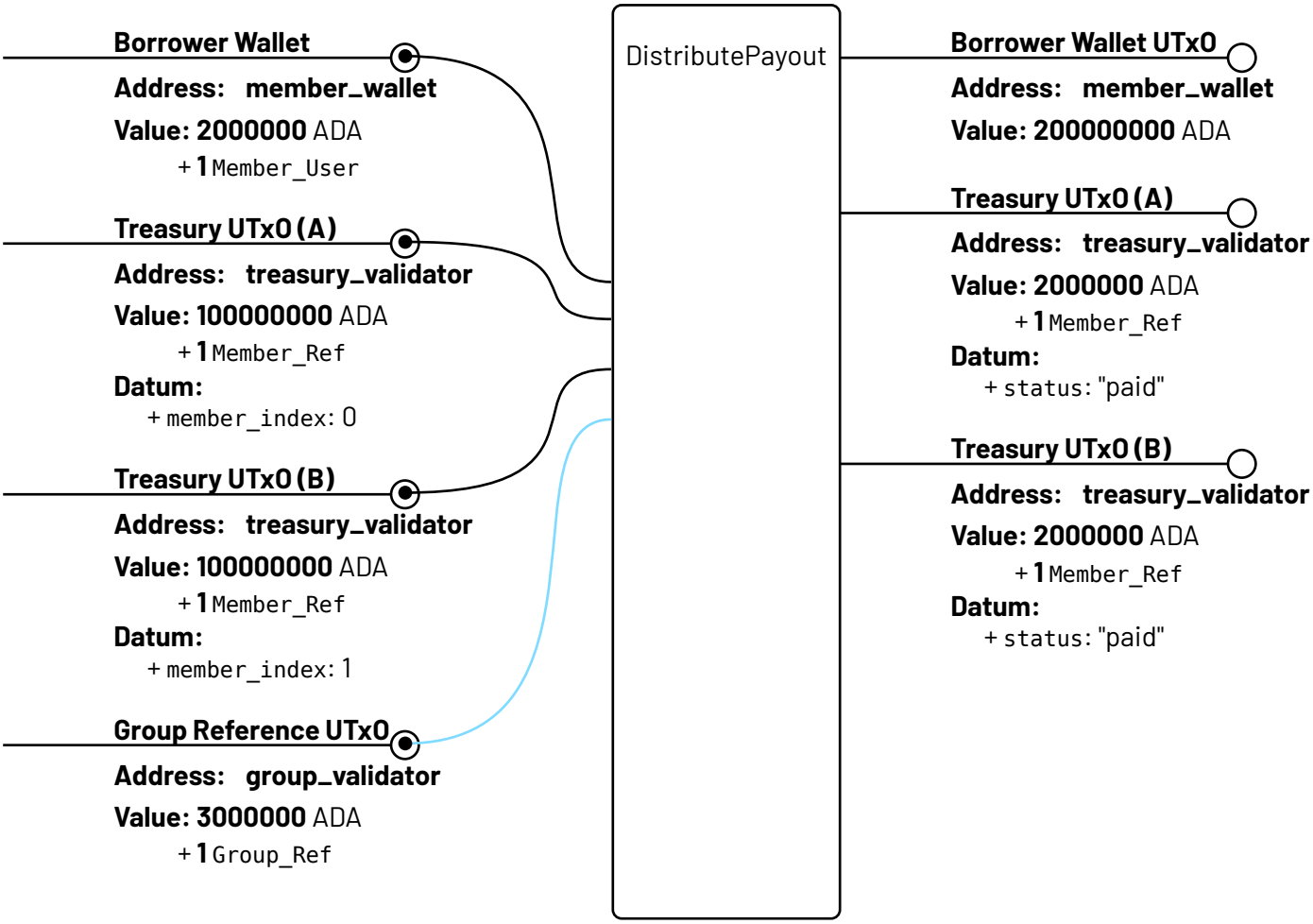
4.3.3.2. Outputs

1. **Member Wallet UTxO**
   - Address: Member's wallet address
   - Value:
     - Minimum ADA
     - Remaining contribution funds (minus any penalties)
     - 1 Account NFT Asset
2. **Treasury Validator UTxO** (if exiting early with penalty)
   - Address: Treasury validator script address
   - Datum:
     - penalty_datum: Metadata indicating the penalty for early exit as listed in Section 3.3.1.4.2
   - Value:
     - Penalty ADA
     - Treasury NFT Asset

Note: If the group is inactive or membership period has ended, the Treasury NFT is burned instead of creating a Penalty UTxO.

4.3.4. Spend :: DistributePayout

This transaction aggregates contributions from multiple members and distributes the lump sum "pot" to the eligible winner of the current rotation interval. It ensures trustless delivery of funds without requiring an administrator to take custody.



Note: Distribute Payout to Borrower

4.3.4.1. Inputs:

1. **Treasury Validator UTxOs (Multiple)**
   - Address: Treasury validator script address
   - Datum:
     - treasury_datum: listed in Section 3.3.1.4.1 (for Member A)
     - treasury_datum: listed in Section 3.3.1.4.1 (for Member B)

    - ...
  - Value:
    - Locked contribution funds per member
    - Treasury NFT Assets

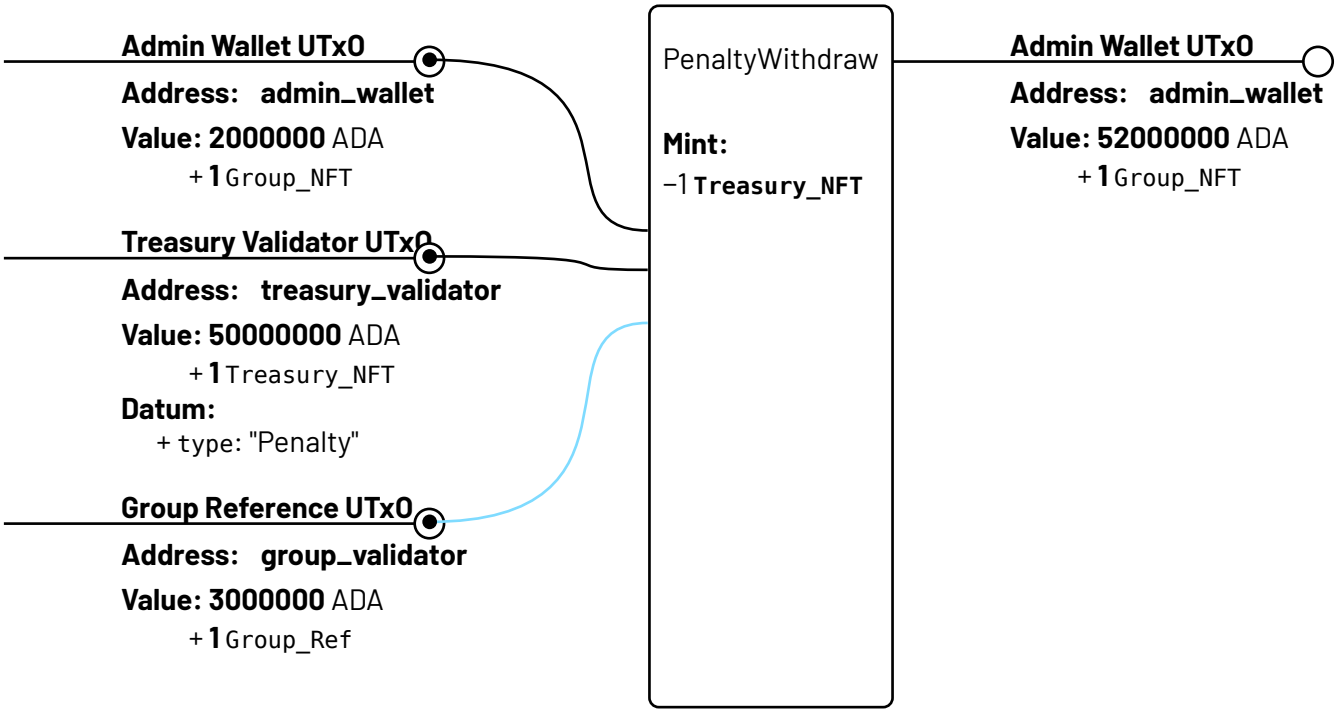
2. **Group Reference UTxO**
   - Address: Group Contract Address
   - Datum:
     - group_datum: listed in Section 3.3.2.3.1
   - Value:
     - 1 Group Reference NFT Asset
     - Minimum Ada

4.3.4.2. Outputs:

1. **Borrower Wallet UTxO**
   - Address: Scheduled Borrower's wallet address
   - Value:
     - Minimum ADA
     - Total collected "Pot" amount (Sum of inputs)
2. **Treasury Validator UTxOs (Multiple)**
   - Address: Treasury validator script address
   - Datum:
     - updated_datum: Metadata reflecting the payout ("Paid" status) for each member
   - Value:
     - Minimum ADA (or remaining balance)
     - Treasury NFT Assets

4.3.5. Spend :: Penalty Withdraw

This transaction allows a group administrator with a Group NFT to unlock penalty funds from the Penalty UTxO, burning the Treasury NFT attached to the UTxO.



Note: Withdraw Penalty Funds

4.3.5.1. Inputs:

1. **Administrator Wallet UTxO**
   - Address: Administrator's wallet address
   - Value:
     - Minimum ADA
     - Group NFT Asset
2. **Treasury Validator UTxO**
   - Address: Treasury validator script address
   - Penalty Datum: as listed in Section 3.3.1.4.2
   - Value:
     - Penalty funds

- Treasury NFT Asset

3. **Group Reference UTxO**
   - Address: Group Contract Address
   - Datum:
     - group_datum: listed in Section 3.3.2.3.1
   - Value:
     - 1 Group Reference NFT Asset
     - Minimum Ada

4.3.5.2. Mints

1. **Treasury Validator**
   - Redeemer: TerminateGroup
   - Value:
     - –1 Treasury NFT Asset

4.3.5.3. Outputs:

1. **Administrator Wallet UTxO**
   - Address: Administrator's wallet address
   - Value:
     - Minimum ADA
     - Withdrawn penalty funds
     - Group NFT Asset
2. **Treasury Validator UTxO**
   - Address: Treasury validator script address
   - Value:
     - Remaining ADA after withdrawal (if any)