ANASTASIA LABS

Payment Subscription Smart Contract

Design Specification

Contents

Payment Subscription Smart Contract

1. Overview

This Payment Subscription Smart Contract is developed using Aiken it is designed to facilitate automated recurring payments between subscribers and merchants on the Cardano blockchain. This contract empowers users to seamlessly set up, manage, and cancel their subscriptions directly from their wallets. It ensures secure and efficient transactions by automating the process of paying subscription fees, updating service metadata, and handling cancellations, all within a decentralized framework.

2. Architecture

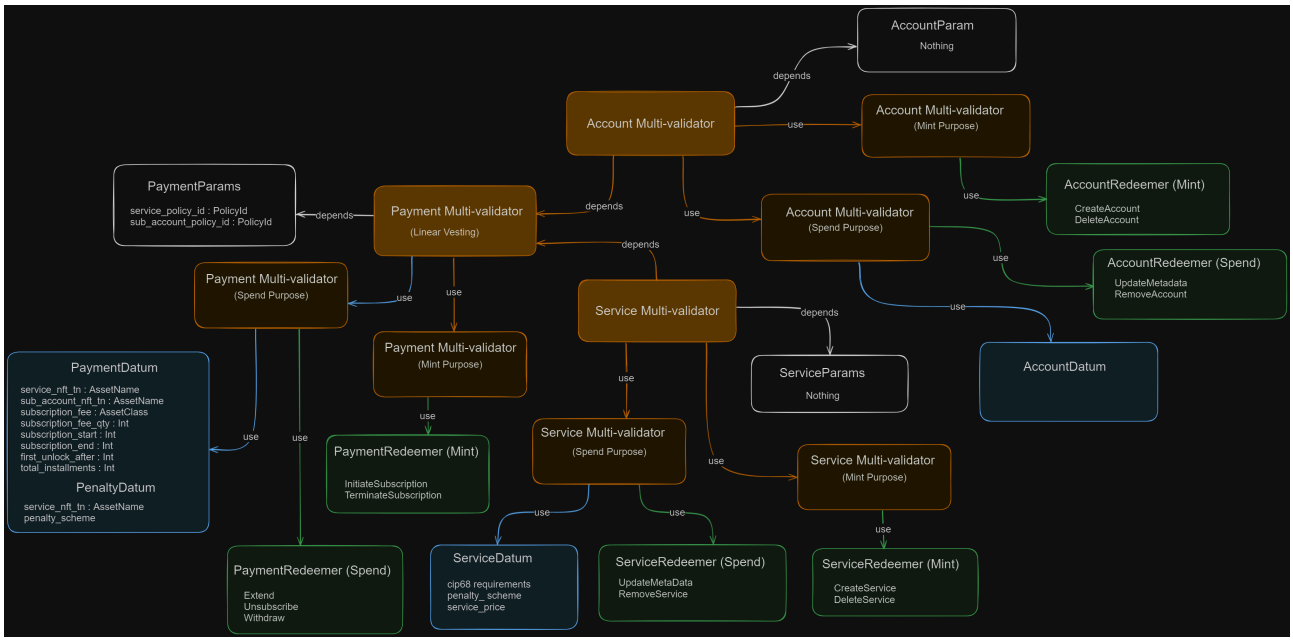

Figure 1: Payment Subscription Architecture

There are three contracts in this subscription system.

1. **Service Contract**

   A multi-validator responsible for creating an initial service by minting a single CIP-68 compliant Service NFT Asset and sending it to the user while sending the reference NFT to the spending endpoint. It also updates the metadata for the user and deletes the service by burning the Service NFT.

2. **Account Contract**

   A multi-validator responsible for creating an account for the user by minting a CIP-68 compliant Account NFT Asset and sending it to the user, while sending the reference NFT to the spending endpoint. It also updates the metadata for the Account and deletes the user account by burning a Account NFT.

3. **Payment Contract**

   This is the core validator. A multi-validator responsible for holding the prepaid subscription fees for a service, renewing a subscription to a service, unsubscribing from a service and withdrawing subscription fees. The contract incorporates a linear vesting mechanism to gradually release subscription fees to the merchant over the subscription period.

3. Specification

3.1. System Actors

1. **Merchant**
   - An entity who interacts with the **Service Contract** to create a service and receive subscription payments for the respective service(s).
   - A user becomes a merchant when they mint a **Service NFT**.
2. **Subscriber**
   - An entity who interacts with the **Account Contract** to create an account and deposit prepaid subscription fees to the **Payment Contract**.
   - A user becomes a subscriber when they mint an **Account NFT** and lock funds to the **Payment Contract**.

3.2. Tokens

1. **Service NFT**
   - Can only be minted by a user when creating a service
   - Burned when the user deletes their service(s) from the system.
   - **TokenName:** Defined in **Service Multi-validator** parameters using a unique token name derived from the transaction ID and output index (using CIP-68 standards).
2. **Account NFT:**
   - Can only be minted by a user when creating an account for the subscription system.
   - Burned when the user deletes their account from the system.
   - A check must be included to verify that there are no payments in the Payment Contract before burning.
   - **TokenName:** Defined in **Account Multi-validator** parameters using a unique token name derived from the transaction ID and output index (using CIP-68 standards).
3. **Payment NFT:**
   - Can only be minted when a subscription fee is paid to the **Payment Contract**
   - Burned when a merchant withdraws the Penalty Fee or Subscription ends.
   - **TokenName:** Defined in **Payment Multi-validator** parameters using a unique token name derived from the transaction ID and output index.

3.3. Smart Contracts

3.3.1. Payment Multi-validator

The Payment Contract is responsible for managing the prepaid subscription fees, validating subscriptions, and ensuring the proper distribution of these fees over the subscription period given the `total_installments`. It facilitates the creation, extension, and cancellation of subscriptions, allowing both subscribers and merchants to interact with the contract in a secure and automated manner. This contract ensures that subscription payments are correctly handled and that any penalties for early cancellation are appropriately enforced.

3.3.1.1. Parameters

- **`service_policy_id`** : Hash of the PolicyId
- **`account_policy_id`** : Hash of the PolicyId

3.3.1.2. Minting Purpose

3.3.1.2.1. Redeemer

- **InitiateSubscription**
- **TerminateSubscription**

3.3.1.2.2. Validation

1. **InitiateSubscription**

   The redeemer allows creating of a new subscription by minting only one unique **Payment Token**.

   - **Validation Steps:**
     - **Presence of Required Inputs:** Validate that the output_reference (the specific UTxO reference) is present in the transaction inputs.
     - **Single Payment NFT Minted:** Ensure that only one unique Payment NFT is minted.
     - **Service Datum Validation:** Validate the datum from the Service Contract as a reference input to ensure it matches the subscription details.
     - **Account NFT Handling:** Ensure that the Account NFT does not go to the script address (i.e., it remains with the subscriber).
     - **Payment NFT Output:** Ensure that the Payment NFT goes back to the script address.
     - **Datum Consistency:** Validate internal consistency of the Payment Datum.
     - **Subscription Details Validation:** Ensure that the subscription details (e.g., interval length, fee amount) match the service parameters.
     - **Locked Amount Validation:** Verify that the correct amount is locked in the contract (e.g., locked_amount >= expected_amount).

2. **TerminateSubscription**
   - **Merchant Ownership Validation:** Validate that the merchant possesses the correct Service NFT.
   - **Single Payment NFT Burned:** Ensure that exactly one Payment NFT is burned during the transaction.
   - **Payment NFT Matching:** Verify that the Payment NFT being burned matches the one from the input.
   - **Service Datum Validation:** Validate the datum from the Service Contract as a reference input.

3.3.1.3. Spend Purpose

This involves the validation logic when spending UTxOs from the Payment Contract. Actions include:

- **Extend Subscription**
- **Merchant Withdraw**
- **Unsubscribe**
- **Subscriber Withdraw**

Each action has specific validation requirements to ensure the integrity and correctness of the subscription process.

3.3.1.3.1. Redeemer

- **Extend**
- **Unsubscribe**
- **MerchantWithdraw**
- **SubscriberWithdraw**

3.3.1.3.2. Validation

1. **Extend**

   Allows subscribers to extend their subscription by locking additional funds.

   - **Validation Steps:**
     - Presence of Required Inputs: Validate that the subscriber's account input and the payment input are present.
     - Service Datum Validation: Validate the datum from the Service Contract as a reference input.
     - Subscription Extension Validity: Ensure that the extension aligns with the service parameters (e.g., maximum subscription length).
     - Fee Increase Validation: Verify that the fee increase corresponds to the extension period.
     - Payment Datum Update: Validate that the new Payment Datum reflects the extended subscription correctly.
     - Token Handling: Ensure that the Payment NFT remains at the script address.

2. **MerchantWithdraw**

   Allows merchants to withdraw accumulated subscription fees corresponding to the elapsed time.

   - **Payment UTxO validation:** (When the datum is a Payment Datum)
     - Merchant Ownership Validation: Validate that the merchant possesses the correct Service NFT.
     - **Service Datum Validation:** Validate the datum from the Service Contract as a reference input.
     - **Withdrawal Amount Calculation:** Calculate the withdrawable amount based on the elapsed subscription period (using linear vesting).
     - **Payment Datum Update:** Update the last_claimed timestamp in the Payment Datum.
     - **Remaining Funds Validation:** Ensure that the remaining funds in the contract are correct.
     - **Minimum ADA Requirement:** Verify that the merchant receives at least the minimum ADA required.
     - **Token Handling:** Ensure that the Payment NFT remains at the script address.

   - **Penalty UTxO validation:** (When the datum is a Penalty Datum)
     - **Merchant Ownership Validation:** Validate that the merchant possesses the correct Service NFT.
     - **Service Datum Validation:** Validate the datum from the Service Contract as a reference input.
     - **Penalty Input Validation:** Ensure that the input UTxO contains the correct Penalty Datum.

- Penalty Fee Validation: Verify that the penalty fee amount in the input matches the expected amount.
- Penalty Withdrawal: Ensure that the penalty fee is transferred to the merchant.
- Payment NFT Burned: Validate that the associated Payment NFT is burned.
- Token Handling: Ensure that the Service NFT remains with the merchant.

3. **Unsubscribe**

   Allows subscribers to cancel their subscription before the end date.

   - **Validation Steps:**
     - Presence of Required Inputs: Validate that the subscriber's account input and the payment input - are present.
     - Service Datum Validation: Validate the datum from the Service Contract as a reference input.
     - Refund Calculation: Calculate any refundable amount based on the elapsed subscription time.
     - Penalty Fee Handling: Apply any penalty fees as defined in the service parameters.
     - Payment Datum Update: Update the contract state to reflect the unsubscription.
     - Penalty Datum Creation: Create a new Penalty Datum representing the penalty to be paid.
     - Token Handling: Ensure that the Payment NFT is transferred to the new penalty UTxO at the script address.

4. **SubscriberWithdraw**

Allows subscribers to withdraw funds if the service becomes inactive (e.g., the merchant has deleted the service).

- **Validation Steps:**
  - Service Inactivity Validation: Validate that the service is inactive.
  - Full Refund: Ensure the subscriber receives a full refund of remaining funds.
  - Payment NFT Handling: Burn the associated Payment NFT.
  - Token Handling: Ensure that the Account NFT remains with the subscriber.

3.3.1.4. Datum

The datum is a sum type with two constructors, representing different states of the subscription.

- **Payment Datum**
- **Penalty Datum**

3.3.1.4.1. Payment datum

Represents the state of an active subscription.

- **`service_nft_tn`:** Service token name encoding UTxO to be consumed when minting the NFT.
- **`account_nft_tn`:** Accont token name encoding UTxO to be consumed when minting the NFT.
- **`subscription_fee`:** AssetClass type for the subscription fee.
- **`total_subscription_fee`:** Total amount of the subscription fee for the entire subscription period.
- **`subscription_start`:** Starting time of the withdrawal period.
- **`subscription_end`:** Expiry time of the subscription.
- **`interval_length`:** Length of an interval,
- **`interval_amount`:** Amount withdrawable at each interval,
- **`num_intervals`:** Number of intervals in the subscription,
- **`last_claimed`:** The last time the merchant withdrew from the contract.
- **`penalty_fee`:** AssetClass type of the penalty.
- **`penalty_fee_qty`:** Amount the merchant wishes to penalies for early withdrawal. This is optional and can be `0`.
- **`minimum_ada`:** Least amount of Ada required to complete the transaction,

3.3.1.4.2. Penalty datum

- **`service_nft_tn`:** Service token name encoding UTxO to be consumed when minting the NFT.
- **`account_nft_tn`:** Account token name encoding the UTxO to be consumed when minting the NFT.
- **`penalty_fee`:** AssetClass type for the amount of fees to be deducted when subscriber cancels the subscription (optional).
- **`penalty_fee_qty`:** Amount of the penalty fees (optional).

3.3.2. Account Multi-validator

Responsible for managing user accounts within the subscription system.

3.3.2.1. Parameter

Nothing

3.3.2.2. Minting Purpose

3.3.2.2.1. Redeemer

- **CreateAccount**
- **DeleteAccount**

3.3.2.2.1.1. Validation

1. **CreateAccount**

   Allows a user to create a new account by minting an Account NFT and a reference NFT.

   - **Validation Steps:**
     - Presence of Required Inputs: Ensure the output_reference is spent in the transaction.
     - Minting Tokens: Validate that exactly one Account NFT and one reference NFT are minted using CIP-68 standards.
     - Account Datum Validation: Validate the Account Datum for correctness (e.g., valid email, phone number).
     - Token Handling: Ensure that the Account NFT does not go to the script address and the reference NFT goes to the script address.

2. **DeleteAccount**

   Allows a user to delete their account, burning the Account NFT and reference NFT.

   A Check That there's no payment for the delete account should be done off-chain.

   - **Validation Steps:**
     - Burning Tokens: Validate that the Account NFT and reference NFT are both burned.

3.3.2.3. Spend Purpose

3.3.2.3.1. Datum

- **`email`:** ByteArray
- **`phone`:** ByteArray
- **`account_created`:** Int

3.3.2.3.2. Redeemer

- **UpdateAccount**
- **RemoveAccount**

3.3.2.3.2.1. Validation

1. **UpdateAccount**

   The redeemer allows for updating the metadata attached to the UTxO sitting at the script address.

   - Validate that the Account UTxO with the Account NFT is being spent.
   - Validate that the metadata of the Reference NFT is updated within acceptable bounds
   - Validate that the UTxO with the Reference NFT of is sent to the spending endpoint.

2. **RemoveAccount**

   The redeemer allows the removal of an account by a subscriber from the subscription system. Must Check That there's no ADA in the payment UTxO in the off-chain code.

   - No Active Subscriptions: Check that there are no active subscriptions in the Payment Contract.
   - Validate that Account NFT and Account Reference NFT is spent.
   - Burning Tokens: Validate that the Account NFT and reference NFT are both burned.
   - Token Handling: Ensure that no outputs contain the reference NFT after burning.

3.3.3. Service Multi-validator

The Service Multi-Validator is responsible for managing services offered by merchants. It allows the creation, update, and deletion of services through specific actions.

3.3.3.1. Parameter

Nothing

3.3.3.2. Minting Purpose

3.3.3.2.1. Redeemer

- CreateService

3.3.3.2.2. Validation

1. **CreateService**

   Allows a merchant to create a new service by minting a Service NFT and a reference NFT.

   - **Validation Steps:**
     - Presence of Required Inputs: Ensure the output_reference is spent in the transaction.
     - Minting Tokens: Validate that exactly one Service NFT and one reference NFT are minted using CIP-68 standards.
     - Service Datum Validation: Validate the Service Datum for correctness (e.g., valid service fee, penalty fee).
     - Token Handling: Ensure that the Service NFT does not go to the script address and the reference NFT goes to the script address.

3.3.3.3. Spend Purpose

Handles operations involving spending UTxOs associated with services.

3.3.3.3.1. Redeemer

- **UpdateService**
- **RemoveService**

3.3.3.3.1.1. Validation

1. **UpdateService**

   Allows a merchant to update their service details.

   - **Validation Steps:**
     - Presence of Required Inputs: Validate that the merchant's service input and the service reference input are present.
     - Service Datum Update: Validate that the updated Service Datum is within acceptable bounds (e.g., service fee changes are within a permitted range).
     - Token Handling: Ensure that the reference NFT is sent back to the script address.

2. **RemoveService**

Allows a merchant to deactivate their service without burning the Service NFT.

- **Validation Steps:**
  - Service Inactivation: Ensure the service's is_active field is set to False.
  - Service Datum Consistency: Validate that other fields in the Service Datum remain unchanged.
  - Token Handling: Ensure that the reference NFT is being spent and is sent back to the script address.

3.3.3.3.2. Datum

- **`service_fee`:** AssetClass type for the amount to pay for a subscription service
- **`service_fee_qty`:** Amount of the funds to pay for a service.
- **`penalty_fee`:** AssetClass type for the amount of fees to be deducted when subscriber cancels the subscription (Optional).
- **`penalty_fee_qty`:** Amount of the penalty fees (Optional).
- **`interval_length`:** Length of an interval for the subscription,
- **`num_intervals`:** Number of intervals in the subscription period (e.g. 12 for monthly withdraws for a year),
- **`minimum_ada`:** Least amount of Ada required to complete the transaction,
- **`is_active`:** Boolean indicating if the service is active

Note: Subscription fees can be based on length of period the subscriber pays for e.g. If they pay for one month, the fees are more than if they pay for 12 months. This introduces the need for a `min_sub_period`.

4. Transactions

This section outlines the various transactions involved in the Payment Subscription Smart Contract on the Cardano blockchain.

4.1. Service Multi-validator

4.1.1. Mint :: CreateService

This transaction creates a new service by minting a Merchant NFT. This transaction is performed by the merchant to indicate that a new service is available.

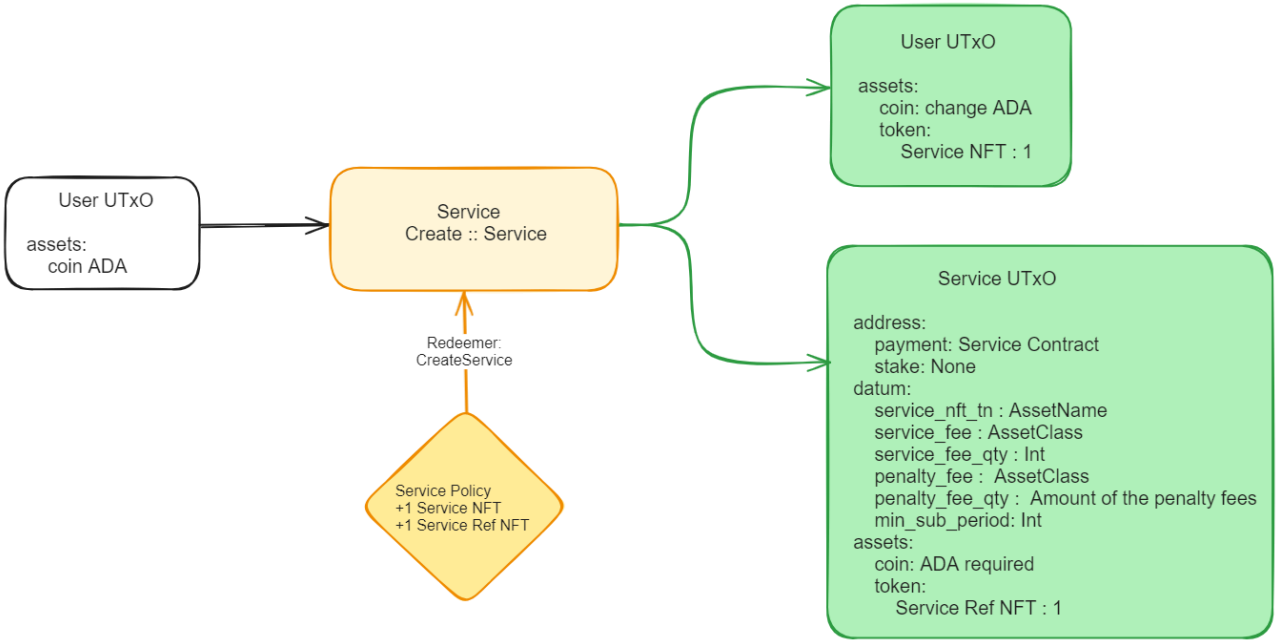

Figure 2: Create Service UTxO diagram

4.1.1.1. Inputs

1. **Merchant Wallet UTxO.**
   - Address: Merchant's wallet address
   - Value:
     - Minimum ADA

- Any ADA required for the transaction.

4.1.1.2. Mints

1. **Service Multi-validator**
   - Redeemer: CreateService
   - Value:
     - +1 Service NFT Asset
     - +1 Reference NFT Asset

4.1.1.3. Outputs

1. **Merchant Wallet UTxO:**
   - Address: Merchant wallet address
     - minimum ADA
     - 1 Service NFT Asset
2. **Service Validator UTxO:**
   - Address: Service Multi-validator Address (Mint)
   - Datum:
     - **`service_nft_tn`:** Service NFT token name encoding UTxO to be consumed when minting the NFT.
     - **`subscription_fee`:** AssetClass type for the subscription fee.
     - **`total_subscription_fee`:** Amount of the subscription fee.
     - **`penalty_fee`**: AssetClass type for the amount of funds to be deducted when subscriber cancels the subscription.
     - **`penalty_fee_qty`**: Amount of the penalty fees.
   - Value:
     - 1 Service Reference NFT Asset

4.1.2. Spend :: UpdateMetaData

This transaction updates the metadata attached to the UTxO at the script address in accordance with CIP-68 standards. It consumes both the Service NFT and the Service Reference NFT, then sends the updated Service NFT to the user's wallet and the updated Reference NFT to the spending endpoint.

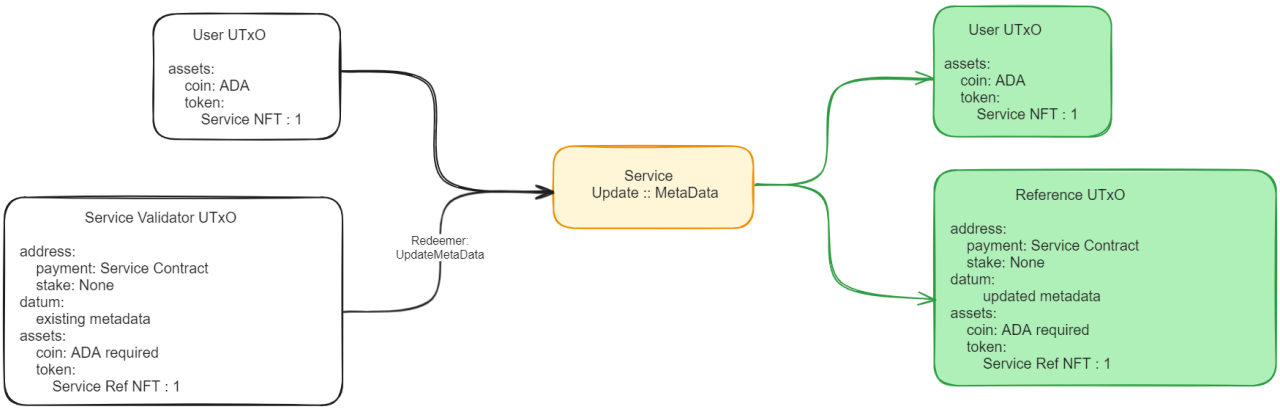

Figure 3: Update Service MetaData UTxO diagram

4.1.2.1. Inputs

1. **Merchant Wallet UTxO**
   - Address: Merchant's wallet address
   - Value:
     - Minimum ADA
     - 1 Service NFT Asset
2. **Service Validator UTxO**
   - Address: Service validator script address
   - Datum:
     - existing_metadata: listed in Section 3.3.3.3.2

.

- Value:
  - Minimum ADA
  - 1 Reference NFT Asset

4.1.2.2. Outputs

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

4.1.3. Spend :: RemoveService

This transaction spends the Reference UTxO and the Service NFT to remove a service from the system.

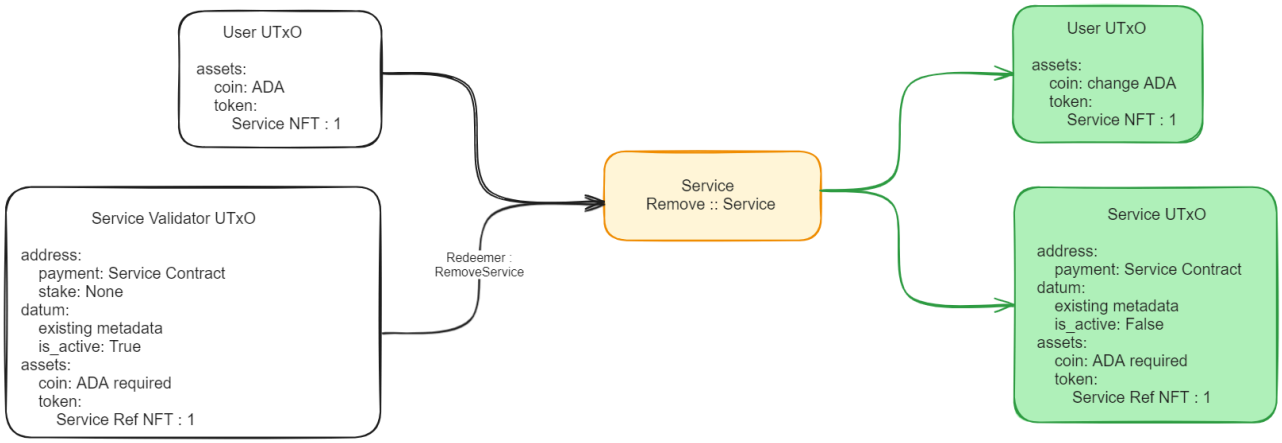

Figure 4: Remove Service UTxO diagram

4.1.3.1. Inputs

1. **Merchant Wallet UTxO**
   - Address: Merchant's wallet address
   - Value:
     - Minimum ADA
     - 1 Service NFT Asset
2. **Service Validator UTxO**
   - Address: Service validator script address
   - Datum:
     - service_metadata: Current metadata listed in Section 3.3.3.3.2.
     - is_active: True
   - Value:
     - Minimum ADA
     - 1 Reference NFT Asset

4.1.3.2. Outputs

1. **Merchant Wallet UTxO**
   - Address: Merchant wallet address
   - Value:
     - Minimum ADA
     - 1 Service NFT Asset
2. **Service Validator UTxO**
   - Address: Service validator script address
   - Datum:
     - service_metadata: Current metadata listed in Section 3.3.3.3.2.
     - is_active: False
   - Value:
     - Minimum ADA
     - 1 Reference NFT Asset

4.2. Account Multi-validator

4.2.1. Mint :: CreateAccount

This endpoint mints a new subscription NFT for a subscriber, establishing a new subscription account.

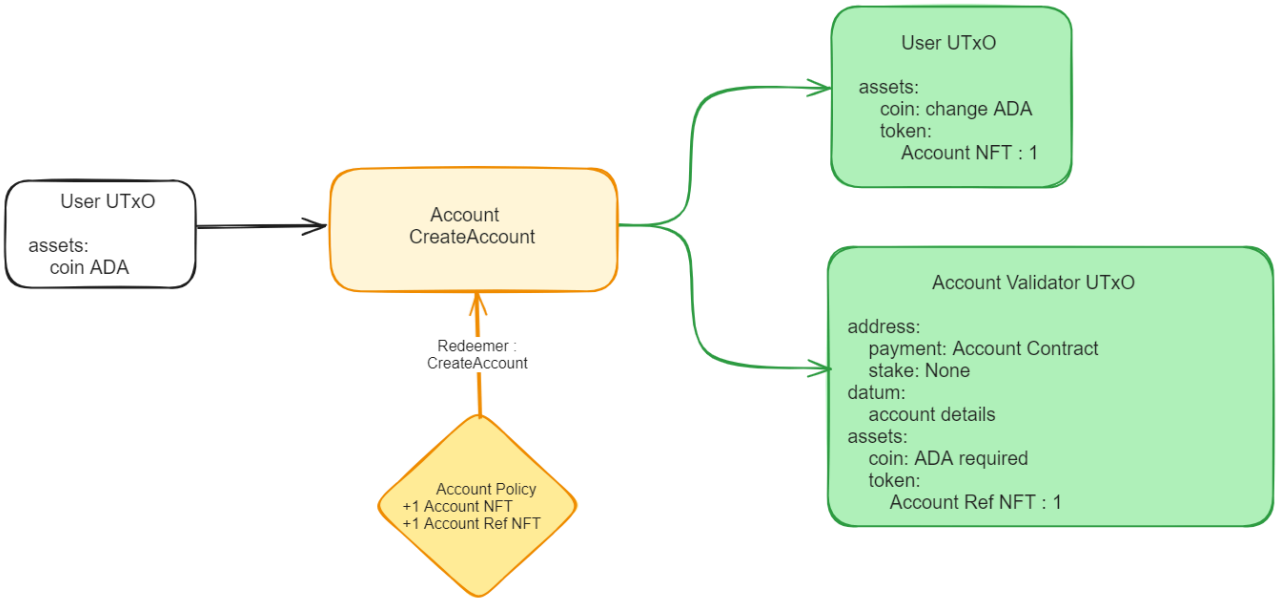

Figure 5: Create Account UTxO diagram

4.2.1.1. Inputs

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

4.2.1.2. Mints

1. **Account Multi-validator**
   - Redeemer: CreateAccount

- Value:
  - +1 Account NFT Asset
  - +1 Reference NFT Asset

4.2.1.3. Outputs

1. **Subscriber Wallet UTxO:**
   - Address: Subscriber wallet address
   - Value:
     - minimum ADA
     - 1 Account NFT Asset
2. **Account Validator UTxO:**
   - Address: Account validator script address
   - Datum:
     - account_details: Arbitrary ByteArray
   - Value:
     - 1 Reference NFT Asset

4.2.2. Mint :: DeleteAccount

This transaction allows a subscriber to burn an Account NFT, effectively removing the user from the subscription system. An off-chain check is required to ensure that there are no pending subscription fees in the Payment UTxO.

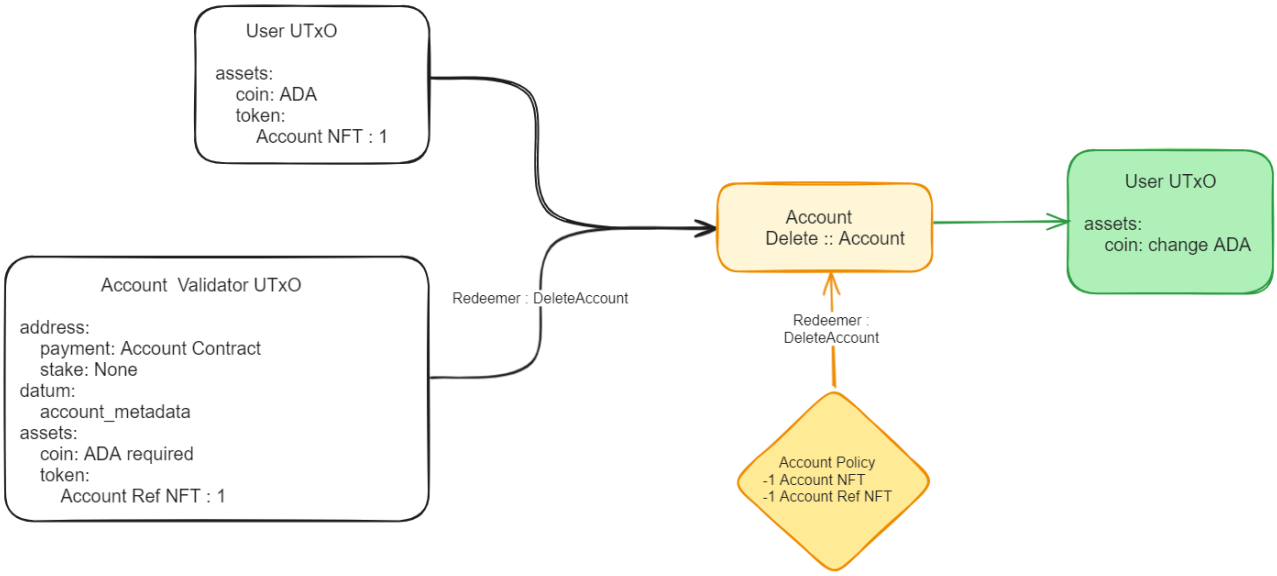

Figure 6: Delete Account UTxO diagram

4.2.2.1. Inputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - Minimum ADA
     - 1 Account NFT Asset
2. **Account Validator UTxO**
   - Address: Account Multi-validator script address
   - Datum:
     - account_details: Arbitrary ByteArray
   - Value:
     - 1 Reference NFT Asset

4.2.2.2. Mints

1. **Account Multi-validator**
   - Redeemer: DeleteAccount
   - Value:
     - −1 Account NFT Asset
     - −1 Reference NFT Asset

4.2.2.3. Outputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - Remaining ADA and other tokens, if any

4.2.3. Spend :: UpdateMetaData

This transaction updates the metadata attached to the Account UTxO at the script address. It consumes both the Account NFT and the Reference NFT, then sends the updated Subscriber NFT to the user's wallet and the updated Reference NFT to the spending endpoint.

Note: The service provider must query UTxOs regularly to facilitate data sync when datum is updated

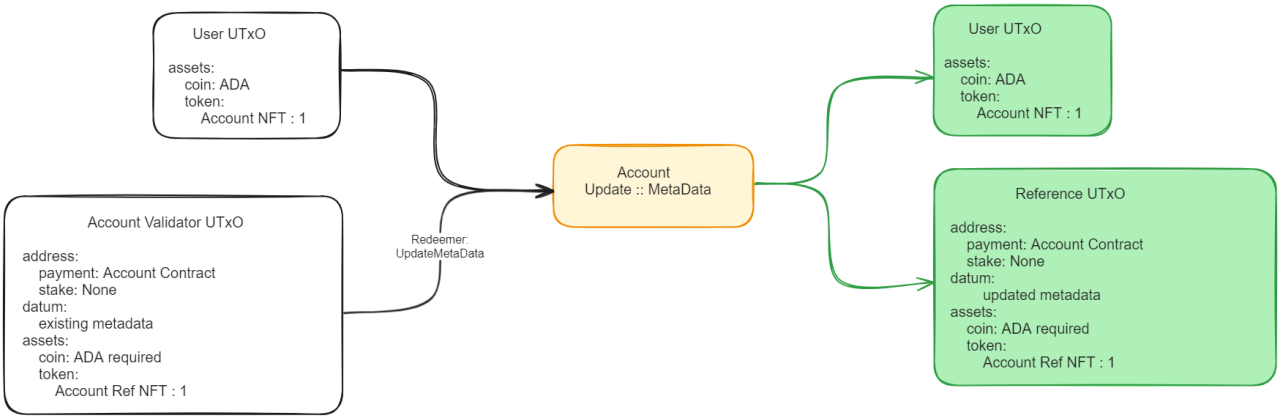

Figure 7: Update Account Metadata UTxO diagram

4.2.3.1. Inputs

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

4.2.3.2. Outputs

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

4.2.4. Spend :: RemoveAccount

This transaction effectively terminates the subscription and removes the subscriber's account from the system by consuming the Account NFT and the Reference NFT.

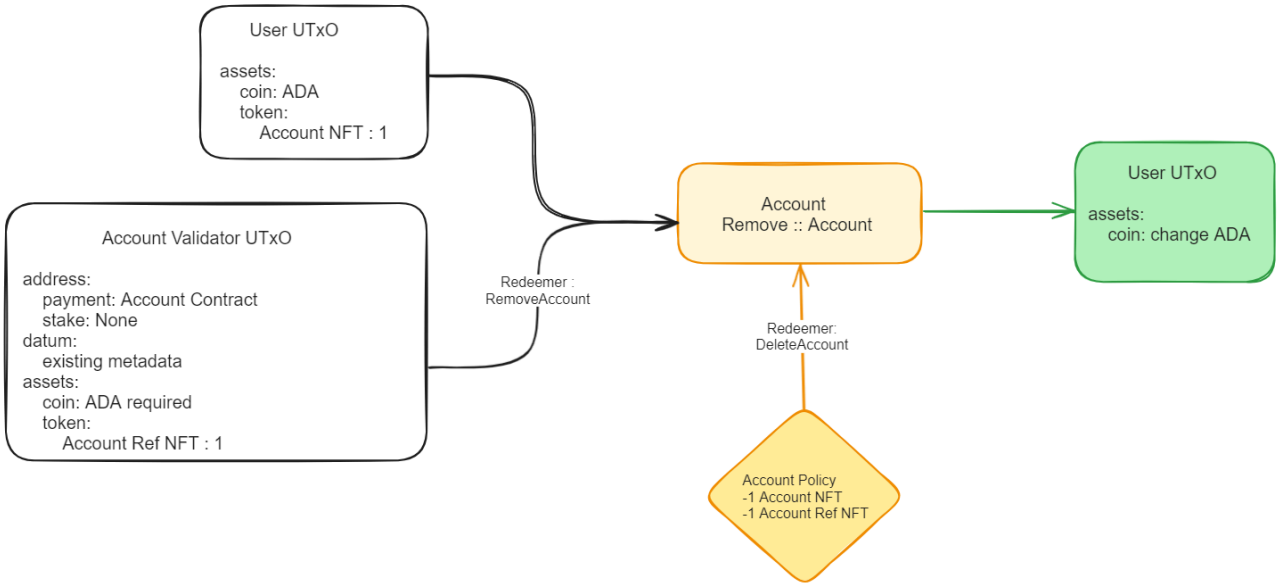

Figure 8: Remove Account Metadata UTxO diagram

4.2.4.1. Inputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - Minimum ADA
     - Account NFT Asset
2. **Account Policy UTxO**

- Address: Account Multi-validator Address (Spend)
- Datum:
  - account_datum: listed in Section 3.3.2.3.1
- Value:
  - Minimum ADA

- 1 Reference NFT Asset

4.2.4.2. Mints

1. **Account Multi-validator**
   - Redeemer: RemoveAccount
   - Value:
     - −1 Account NFT Asset
     - −1 Reference NFT Asset

4.2.4.3. Outputs

1. **Subscriber UTxO**
   - Address: Subscriber wallet address
   - Value:
     - Minimum ADA (remaining after burning the NFT)

4.3. Payment Multi-validator

4.3.1. Mint :: InitiateSubscription

This transaction occurs when a Subscriber locks funds in the Payment validator script address.

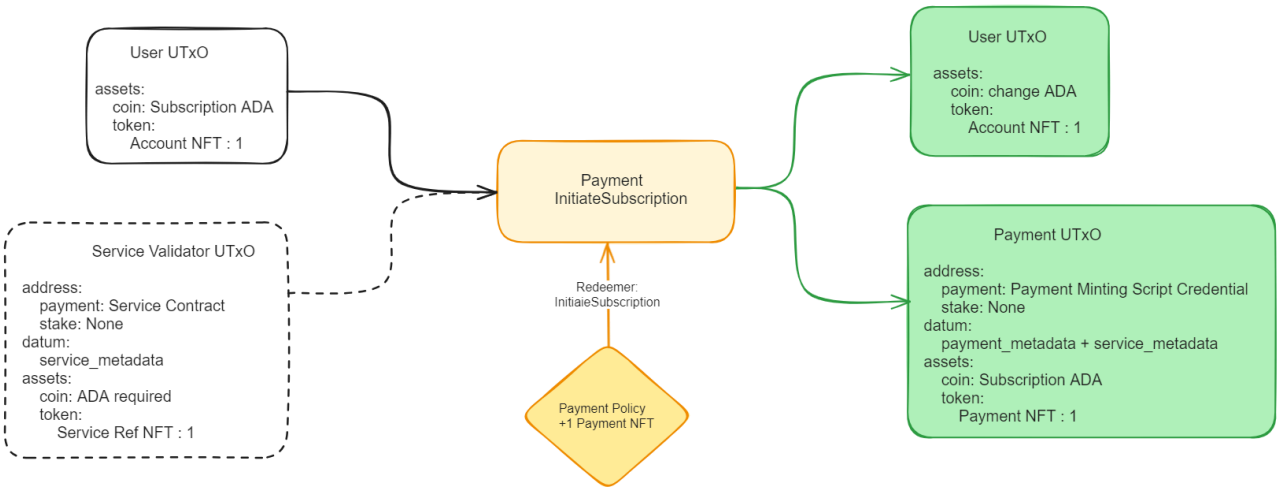

Figure 9: Initiate Subscription UTxO diagram

4.3.1.1. Inputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - 100 ADA: Amount of ADA to Add to the Payment Contract.
     - 1 Account NFT Asset

4.3.1.2. Mints

1. **Service Multi-validator**
   - Redeemer: InitiateSubscription
   - Value:
     - +1 Payment NFT Asset

4.3.1.3. Outputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - Change ADA
     - 1 Account NFT Asset
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - payment datum as listed in Section 3.3.1.4.1
   - Value:
     - 100 ADA: Subscription funds to be withdrawn by merchant
     - 1 Payment NFT Asset

4.3.2. Spend :: Extend

This transaction allows anyone to extend their subscription period by adding more funds to the Payment contract to cover additional time.

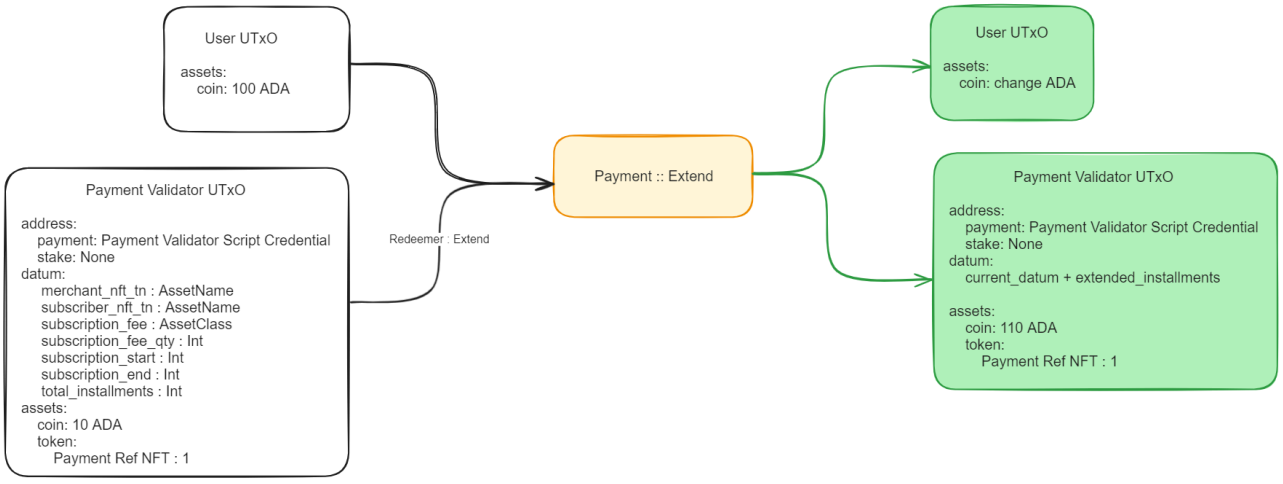

Figure 10: Extend Plan UTxO diagram

4.3.2.1. Inputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - 100 ADA: Amount of ADA to Add to the Contract to extend the payment plan
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - datum listed in Section 3.3.1.4.1
   - Value:
     - 10 ADA: Original amount of ADA before Extending
     - 1 Payment NFT Asset

4.3.2.2. Outputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - –100 ADA
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - datum listed in Section 3.3.1.4.1 + extended installments
   - Value:
     - 110 ADA: Increased ADA to cover the extended subscription period
     - 1 Payment NFT Asset

4.3.3. Spend :: Unsubscribe

This transaction allows the owner of an Account NFT to unsubscribe from a particular service by spending a Payment UTxO, unlocking the remainig pre-paid subscription fee to their own wallet address and creating a Penalty UTxO.

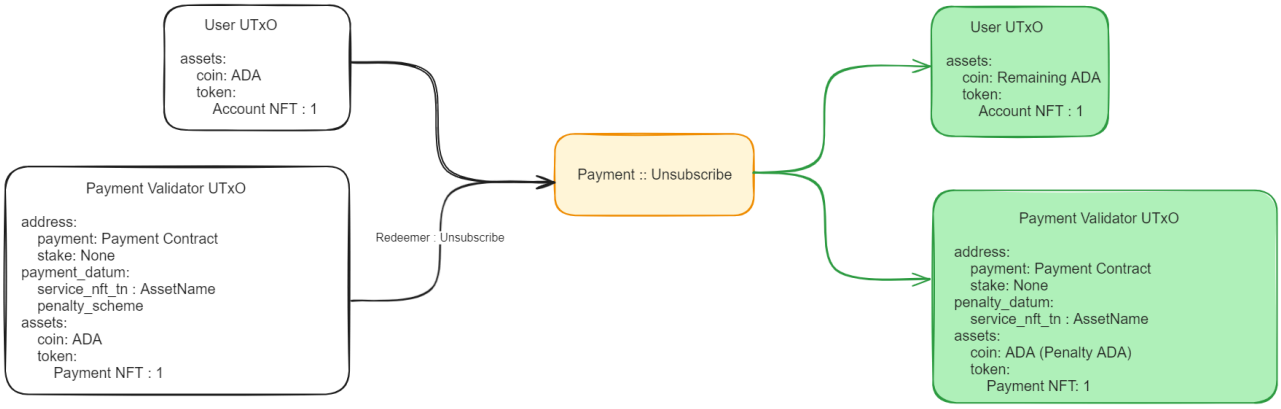

Figure 11: Unsubscribe UTxO diagram

4.3.3.1. Inputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - Minimum ADA
     - 1 Account NFT Asset
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - current_datum: Current payment metadata listed in Section 3.3.1.4.1
   - Value:
     - Minimum ADA
     - Reference NFT Asset

4.3.3.2. Outputs

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - Minimum ADA
     - Unspent portion of the subscription fee (minus any penalties)
     - 1 Account Token Asset
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - penalty_datum: Metadata indicating the penalty for early unsubscription in Section 3.3.1.4.2
   - Value:
     - Penalty ADA
     - Reference NFT Asset

4.3.4. Spend :: Withdraw

This transaction allows anyone with a Service NFT to unlock subscription funds from the Payment UTxO in repect to the specified subscription start and end dates.

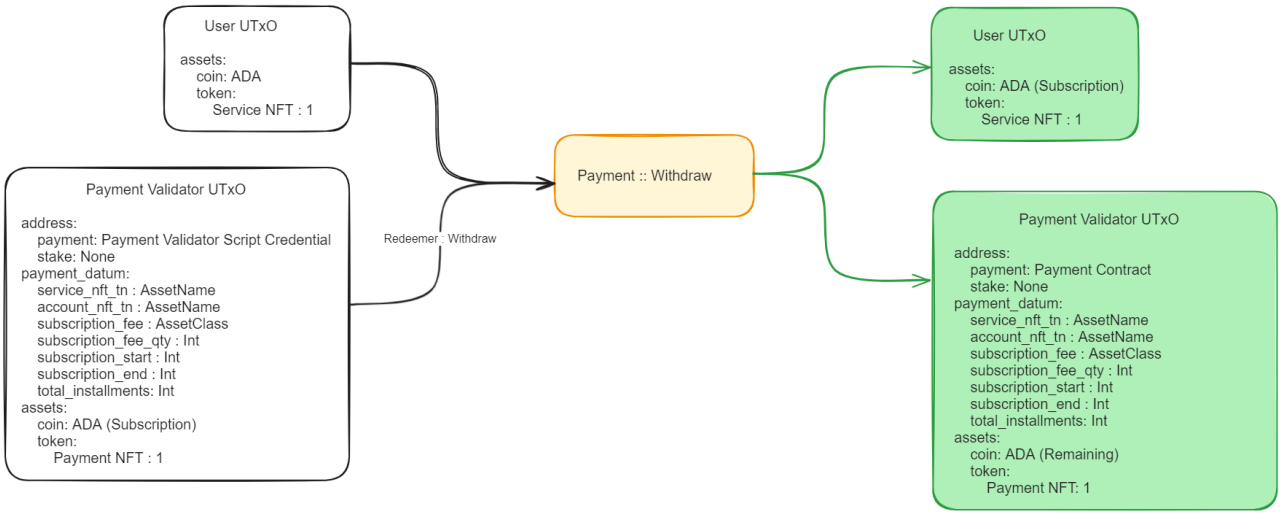

Figure 12: Merchant Withdraw UTxO diagram

4.3.4.1. Inputs:

1. **Merchant Wallet UTxO**
   - Address: Merchant's wallet address

total_subscription_fee
- Value:
  - Minimum ADA
  - 1 Service NFT Asset

1. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - payment_datum: listed in Section 3.3.1.4.1
   - Value:
     - Minimum ADA
     - 1 Payment NFT Asset

4.3.4.2. Outputs:

+ **Merchant Wallet UTxO**

- Address: Merchant's wallet address
- Value:
  - Minimum ADA
  - Withdrawn subscription fee for the installment
  - 1 Service NFT Asset

1. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Datum:
     - updated_datum: Metadata reflecting the withdrawal
   - Value:
     - Remaining ADA after withdrawal
     - 1 Payment NFT Asset

4.3.5. Spend :: Penalty Withdraw

This transaction allows anyone with a Service NFT to unlock penalty funds associated to the service from the Penalty UTxO, in turn burning the Payment NFT attached to the UTxO.

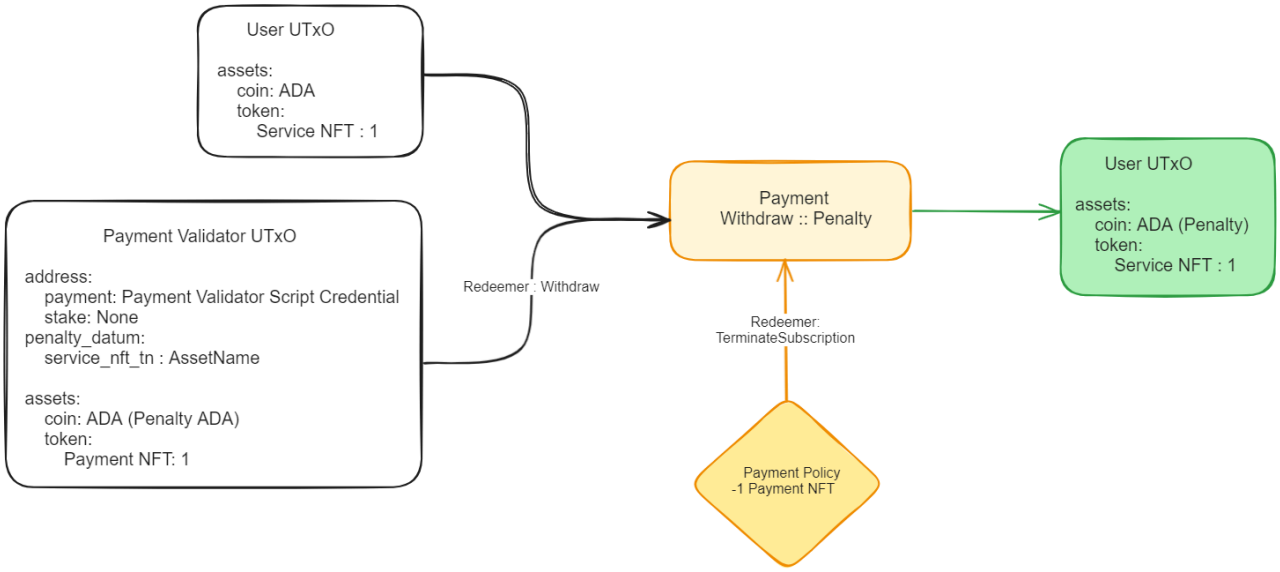

Figure 13: Penalty Withdraw UTxO diagram

4.3.5.1. Inputs:

1. **Merchant Wallet UTxO**
   - Address: Merchant's wallet address
   - Value:
     - Minimum ADA
     - Service NFT Asset
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Penalty Datum:
     - service-nft-tn: Service AssetName
   - Value:
     - Minimum ADA

- Payment NFT Asset

4.3.5.2. Mints

1. **Service Multi-validator**
   - Redeemer: TerminateSubscription
   - Value:
     - −1 Payment NFT Asset

4.3.5.3. Outputs:

1. **Merchant Wallet UTxO**
   - Address: Merchant's wallet address
   - Value:
     - Minimum ADA
     - Withdrawn subscription fee for the installment
     - Service NFT Asset
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Value:
     - Remaining ADA after withdrawal

4.3.6. Spend :: Subscriber Withdraw

This transaction allows anyone with an Accoun NFT to unlock subscription funds from the Payment UTxO only if the status in the ServiceDatum is inactive.

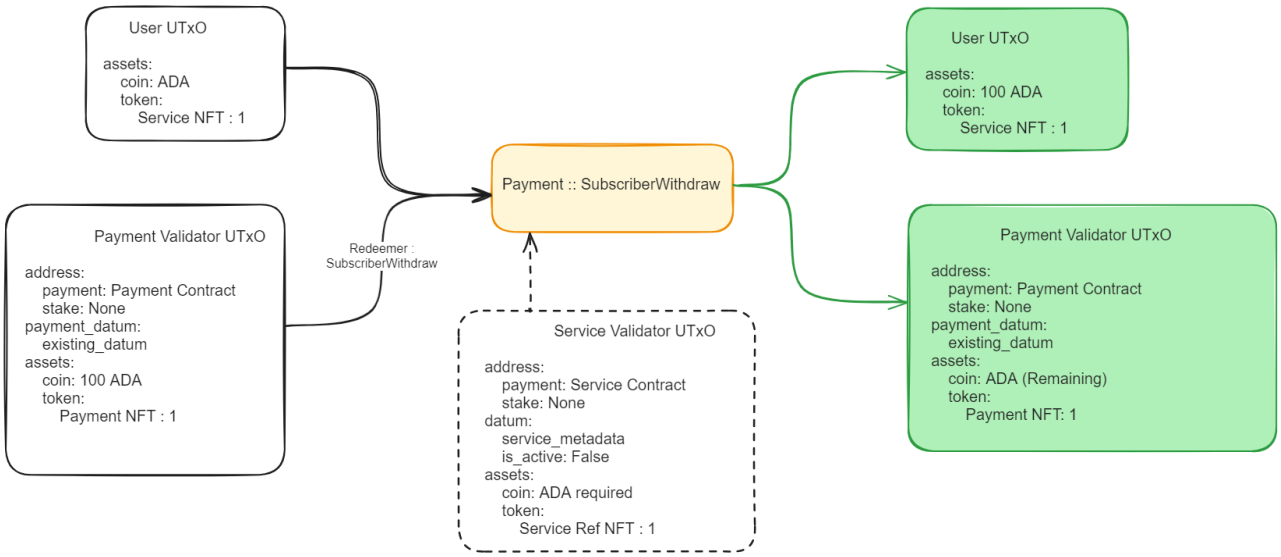

Figure 14: Subscriber Withdraw UTxO diagram

4.3.6.1. Inputs:

1. **Subscriber Wallet UTxO**
   - Address: Subscriber's wallet address
   - Value:
     - ADA
     - Account NFT Asset
2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Payment Datum:
     - existing_datum
   - Value:
     - Subscription ADA
     - Payment NFT Asset

4.3.6.2. Outputs:

1. **Subscriber Wallet UTxO**

- Address: Subscriber's wallet address
- Value:
  - Withdrawn Subscription ADA
  - Account NFT Asset

2. **Payment Validator UTxO**
   - Address: Payment validator script address
   - Payment Datum:
     - existing_datum
   - Value:
     - Remaining ADA after subscriber withdrawal