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

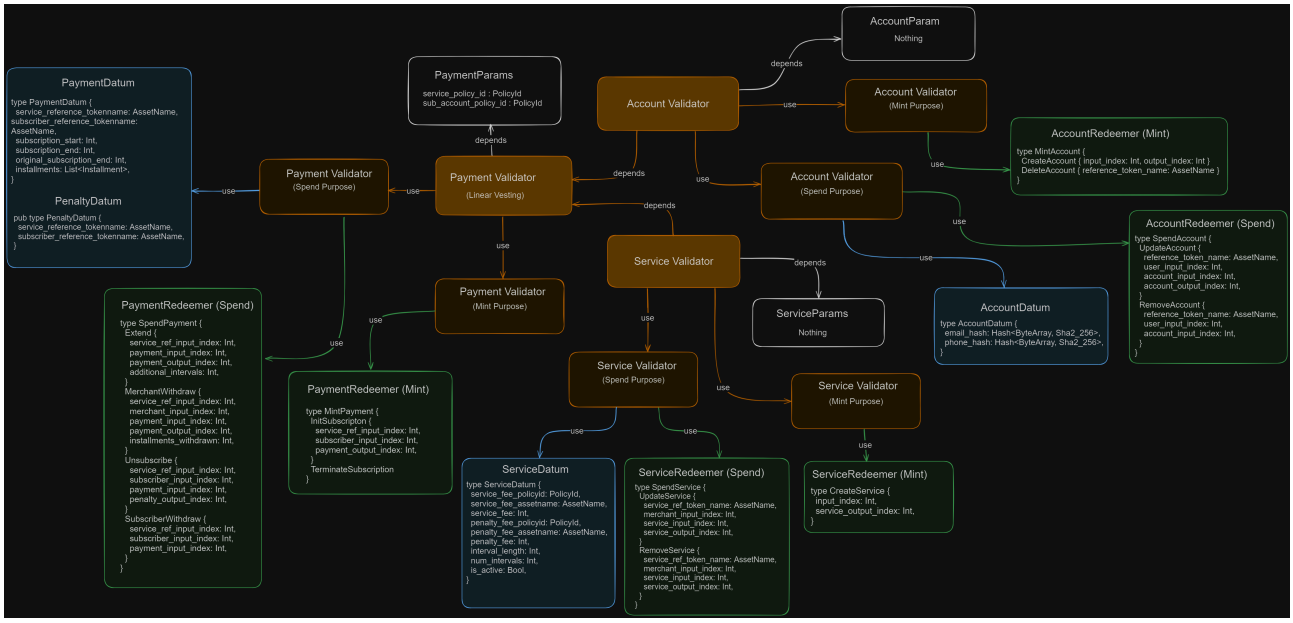

Figure 1: Payment Subscription Architecture

There are three contracts in this subscription system.

1. **Service Contract**

   A 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 allows the wallet address with the user(merchant) NFT to update the metadata as well as deactivate the service.

2. **Account Contract**

   A 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 Account contract address. It also alows the wallet address containing the user(subscriber) NFT to update the metadata for the Account and deletes the user account by burning the Account NFTs.

3. **Payment Contract**

   This is the core validator. The 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 a subscriber specified subscription period.

3. Specification

3.1. System Actors

1. **Merchant**

   This is the 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 and receive a Service NFT in their wallet.

2. **Subscriber**

   This is the 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 and burned when the user deletes their service(s) from the system.

   - **TokenName:** Defined in Service validator while creating a Service with the transaction ID and the output index and prefix for the specific token i.e prefix_100 for the reference NFT and prefix_222 for the user NFT.

2. **Account NFT:**

   Can only be minted by a user when creating an account for the subscription system and 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 validator while creating an Account with the transaction ID and the output index and prefix for the specific token i.e prefix_100 for the reference NFT and prefix_222 for the user NFT.

3. **Payment NFT:**

   Can only be minted when a subscription fee is paid to the Payment Contract and burned when a subscriber exits the system.

   - **TokenName:** Defined in Payment validator with a static string i.e. “payment-subscription”.

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 **`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 may be 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

- ```
  InitSubscripton {
    service_ref_input_index: Int,
    subscriber_input_index: Int,
    payment_output_index: Int,
  }
  ```
- **`TerminateSubscription`**

3.3.1.2.2. Validation

1. **InitSubscripton**

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

   - A reference input must provide the Service datum from the Service Contract.
   - The subscriber's transaction input must contain the correct Account NFT (derived from the subscriber reference token name).
  ```

- The payment output must be sent to the Payment Script's address and contain a Payment datum that is consistent with the Service datum.
- Exactly one Payment Token (with token name "subscription") is minted.
- Ensure that the User NFT doesn't go to the Script
- Ensure Payment token goes back to the script

2. **TerminateSubscription**
   - The redeemer must burn exactly one Payment Token (i.e. a single token with the token name "subscription" is burned).

3.3.1.3. Spend Purpose

3.3.1.4. Datum

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

3.3.1.4.1. Payment datum

- **`service_reference_tokenname: AssetName`** – Links to the Service Contract.
- **`subscriber_reference_tokenname: AssetName`** – Identifies the subscriber's Account NFT.
- **`subscription_start: Int`** – The start time of the subscription.
- **`subscription_end: Int`** – The current expiry time of the subscription.
- **`original_subscription_end:`** The originally agreed subscription end time.
- **`installments:`** List of Installment – Each installment specifies when and how much of the fee becomes withdrawable.
  - **`Installment`**:
    - **`claimable_at : Int`** – Time after which the installment can be claimed.
    - **`claimable_amount : Int`** – The amount available for withdrawal at that time.

3.3.1.4.2. Penalty datum

- **`service_reference_tokenname: AssetName`** – Links to the Service Contract.
- **`subscriber_reference_tokenname: AssetName`** – Identifies the subscriber's Account

3.3.1.5. Redeemer

- ```
  Extend {
    service_ref_input_index: Int,
    payment_input_index: Int,
    payment_output_index: Int,
    additional_intervals: Int,
  }
  ```
- ```
  MerchantWithdraw {
    service_ref_input_index: Int,
    merchant_input_index: Int,
    payment_input_index: Int,
    payment_output_index: Int,
    installments_withdrawn: Int,
  }
  ```
- ```
  Unsubscribe {
    service_ref_input_index: Int,
    subscriber_input_index: Int,
    payment_input_index: Int,
    penalty_output_index: Int,
  }
  ```
- ```
  SubscriberWithdraw {
    service_ref_input_index: Int,
    subscriber_input_index: Int,
    payment_input_index: Int,
  }
  ```

3.3.1.6. Validation

1. **Extend**

   This redeemer/ endpoint will allow anyone to increase the subscription funds by locking the funds in the Payment Contract.

   - The Payment UTxO being extended must be identified by its output reference matching the provided reference.
   - A Service datum must be supplied as a reference input (at **`service_ref_input_index`**), ensuring that the Service NFT (using the **`service_reference_tokenname`**) is present.
   - The output UTxO (at payment_output_index) must remain at the Payment Script's address and include an updated Payment datum.
   - The new Payment datum is validated by comparing it with the current Payment datum to check that the additional locked funds correspond to the specified **`additional_intervals`**.

2. **MerchantWithdraw**

   The redeemer has two variants based on the datum. It allow anyone with a merchant to withdraw funds from the Payment UTxO or Penalty UTxO of their Service depending on the respective Payment and Penalty datums.

   1. **Payment**
      - The merchant must provide an input (at **`merchant_input_index`**) that proves ownership of the Service NFT (derived from the **`service_reference_tokenname`**).
      - The Payment UTxO (from payment_input_index) must have a valid Payment datum which is validated against the service datum passed from **`service_reference_tokenname`**
      - Implement linear vesting for fund release by:
        - Dropping the first **`installments_withdrawn`** elements from the original installments list to form the new Payment datum.
        - Verifying that the difference in the service fee value (calculated from the UTxO's value) between the input and output does not exceed the sum of the **`claimable_amount`** of installments that are past their **`claimable_at`** time.
      - The output UTxO (at **`payment_output_index`**) must be sent to the Payment Script's address.

   2. **Penalty**
      - If a penalty is being applied, the Payment Token must be burned (verified by checking that the mint value includes a burn of one token with the specific Payment NFT token name).
      - The merchant must similarly prove ownership of the Service NFT in one of the inputs (**`merchant_input_index`**).

3. **Unsubscribe**

   The redeemer will allow anyone with an Account NFT to spend an Account UTxO to unlock funds back to their address.

   - The subscriber must provide an input (at **`subscriber_input_index`**) containing the appropriate Account NFT.
   - The Payment UTxO being spent (from **`payment_input_index`**) must carry a valid Payment datum.
   - A Service datum is provided as a reference input to verify service conditions.
     - The decision branch is based on the subscription's timing:
       - **Without Penalty:** If the current time is past the **`original_subscription_end`** or if the Service is inactive, the Payment Token is burned.

- **With Penalty:** If unsubscribing early (active service), 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 Service datum.

4. **SubscriberWithdraw**

   The redeemer will allow anyone with an Account NFT to withdraw funds from the Payment validator.

   - The subscriber's input (at subscriber_input_index) must contain the correct Account NFT.
   - The Payment UTxO (from payment_input_index) must have a valid Payment datum.
   - The Service datum (from the reference input at service_ref_input_index) must indicate that the Service is inactive.
   - The Payment Token is burned (ensuring exactly one token with token name "subscription" is burned).

3.3.2. Service Validator

The Service Multi-validator is responsible for, creating, updating and removing a service.

3.3.2.1. Parameter

Nothing

3.3.2.2. Minting Purpose

3.3.2.2.1. Redeemer

- CreateService

3.3.2.2.2. Validation

1. **CreateService**

   The redeemer allows creating of a new subscription sevice by minting only one unique CIP-68 compliant Service 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 **`service_output_index`** must be sent to the Service Script's address.
   - The output must contain a Service datum with the following requirements:
     - **service_fee**: Must be greater than 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).
     - **is_active**: Must be set to true.

3.3.2.3. Spend Purpose

3.3.2.3.1. Datum

- **`service_fee_policyid: PolicyId`** The PolicyId governing the asset used for the service fee.

- **`service_fee_assetname: AssetName`** The AssetName of the service fee.
- **`service_fee: Int`** An Int representing the fee amount for the service.
- **`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 subscriber cancels early.
- **`interval_length: Int`** An Int defining the duration of one subscription interval.
- **`num_intervals: Int`** An Int representing the total number of intervals in the subscription period.
- **`is_active: Bool`** A Bool indicating whether 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`.

3.3.2.3.2. Redeemer

- ```
  UpdateService {
    service_ref_token_name: AssetName,
    merchant_input_index: Int,
    service_input_index: Int,
    service_output_index: Int,
  }
  ```
- ```
  RemoveService {
    service_ref_token_name: AssetName,
    merchant_input_index: Int,
    service_input_index: Int,
    service_output_index: Int,
  }
  ```

3.3.2.3.2.1. Validation

1. **UpdateService**

   This redeemer endpoint allows anyone to update the metadata attached to a Service UTxO.

   - A Service UTxO containing the Service NFT must be provided (at service_input_index) with its output reference matching the provided reference.
   - A merchant input (at merchant_input_index) must be present to prove ownership of the Service NFT (derived from service_ref_token_name).
   - The output at service_output_index must be sent to the Service Script's address and must include an updated Service datum.
   - Validate that the metadata of the Reference NFT token is updated within acceptable bounds.

- Metadata changes must be within acceptable bounds (for example, service fee adjustments limited to within +/-10%).
- The reference token must be spent back to its own address, ensuring that the Service NFT remains intact.

2. **RemoveService**

   This redeemer endpoint allows a merchant to remove a service from the subscription system.

   - The transaction must include two script inputs:
     - One input containing the Service UTxO with the Service NFT (at **`service_input_index`**).
     - A merchant input (at **`merchant_input_index`**) proving ownership of the Service NFT.
   - Two script outputs must be produced, with one of them (at **`service_output_index`**) sent to the Service Script's address.
   - The output Service datum must indicate that the service is inactivated by setting is_active to false.
   - The Service NFT must still be present in the output to maintain correct state tracking.

3.3.3. Account Validator

The Account Multi-validator handles the creation, update, and removal of subscriber accounts.

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 of a new subscription service account by minting only 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 subscriber account by burning the associated Account Tokens.

A Check That there's no payment for the delete account 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 subscriber'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 subscriber'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 subscriber 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 subscriber from the subscription system.

   Must Check That there's no ADA in the payment UTxO 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 Payment Subscription Smart Contract on the Cardano blockchain.

4.1. Service 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.

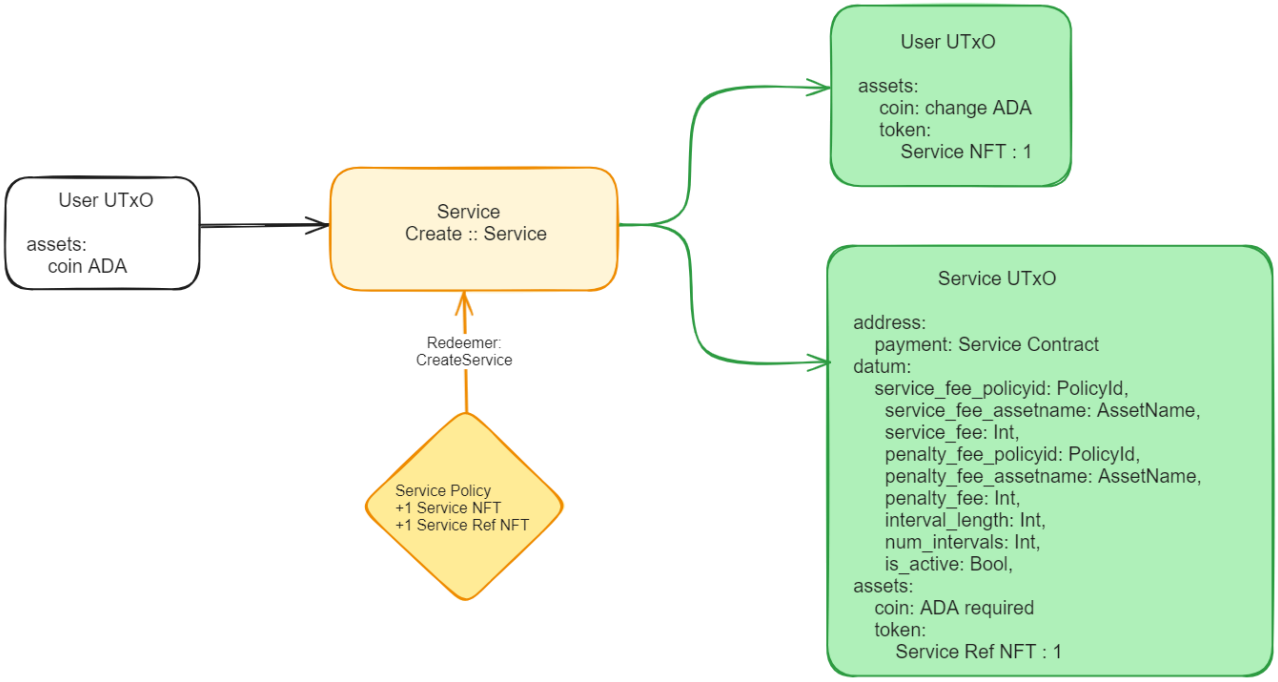

Figure 2: Create Service UTxO diagram

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_fee_policyid: PolicyId`** The PolicyId governing the asset used for the service fee.
     - **`service_fee_assetname: AssetName`** The AssetName of the service fee.
     - **`service_fee: Int`** An Int representing the fee amount for the service.
     - **`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 subscriber cancels early.
     - **`interval_length: Int`** An Int defining the duration of one subscription interval.
     - **`num_intervals: Int`** An Int representing the total number of intervals in the subscription period.
     - **`is_active: Bool`** A Bool indicating whether the service is active.
   - Value:
     - 1 Service Reference NFT Asset

4.1.2. Spend :: UpdateService

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.

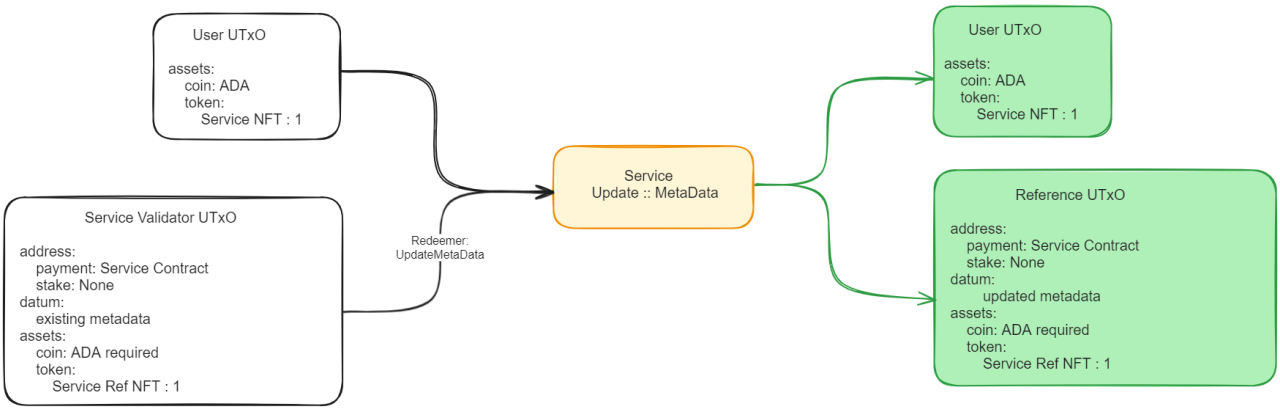

Figure 3: Update Service MetaData UTxO diagram

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 3.3.2.3.1

.

- 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.

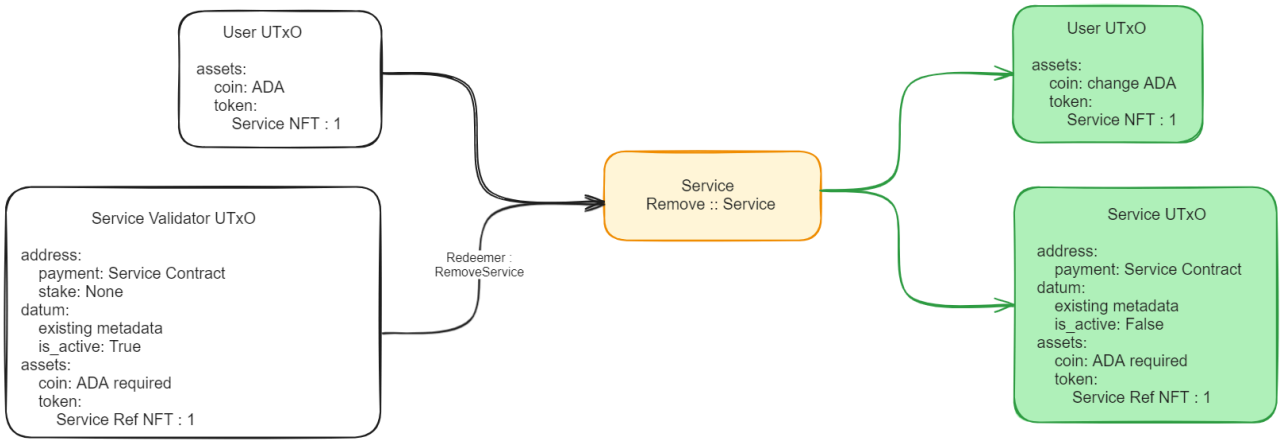

Figure 4: Remove Service UTxO diagram

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 3.3.2.3.1.
     - 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 3.3.2.3.1.
     - is_active: False
   - Value:
     - Minimum ADA
     - 1 Reference NFT Asset

4.2. Account Validator

4.2.1. Mint :: CreateAccount

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

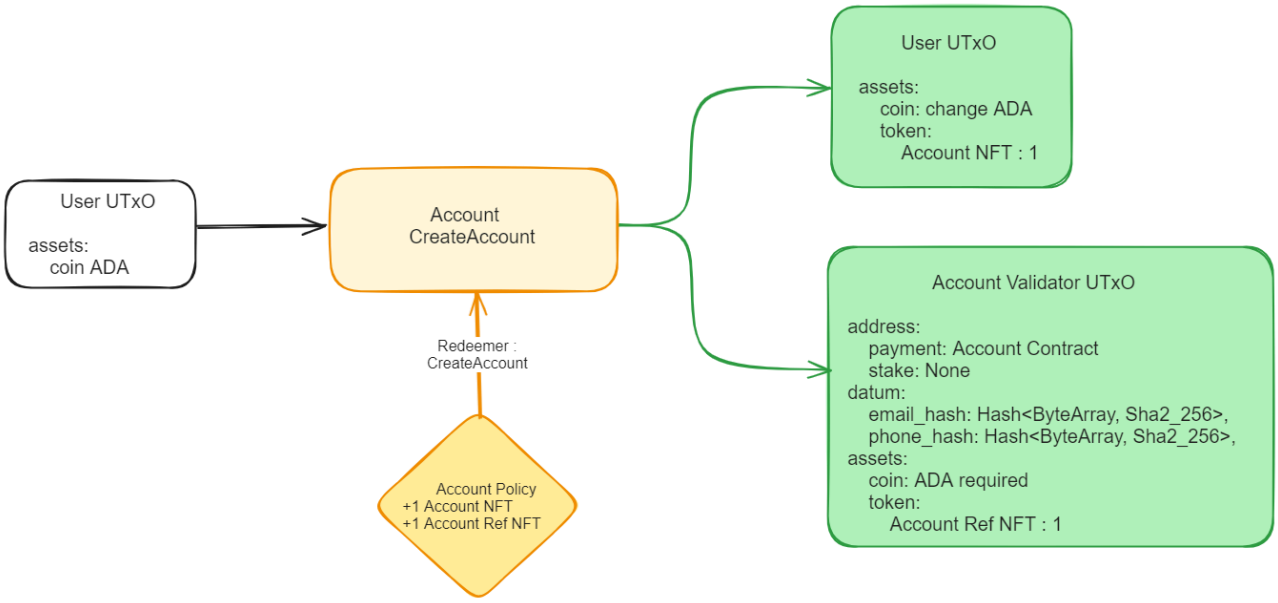

Figure 5: Create Account UTxO diagram

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

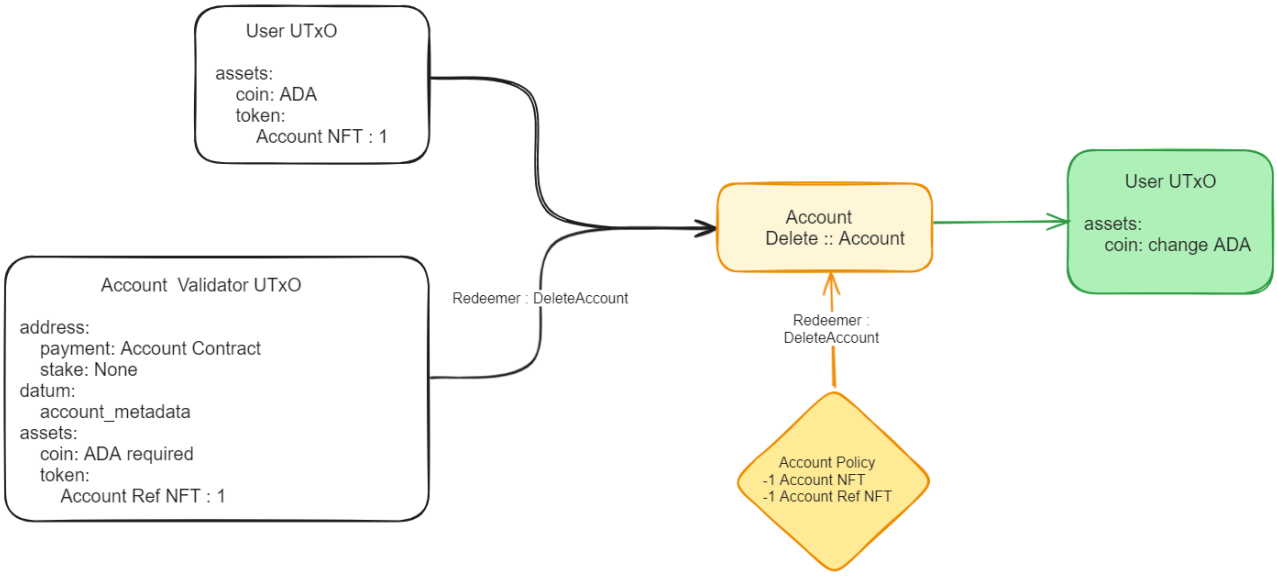

Figure 6: Delete Account UTxO diagram

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 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 :: UpdateAccount

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

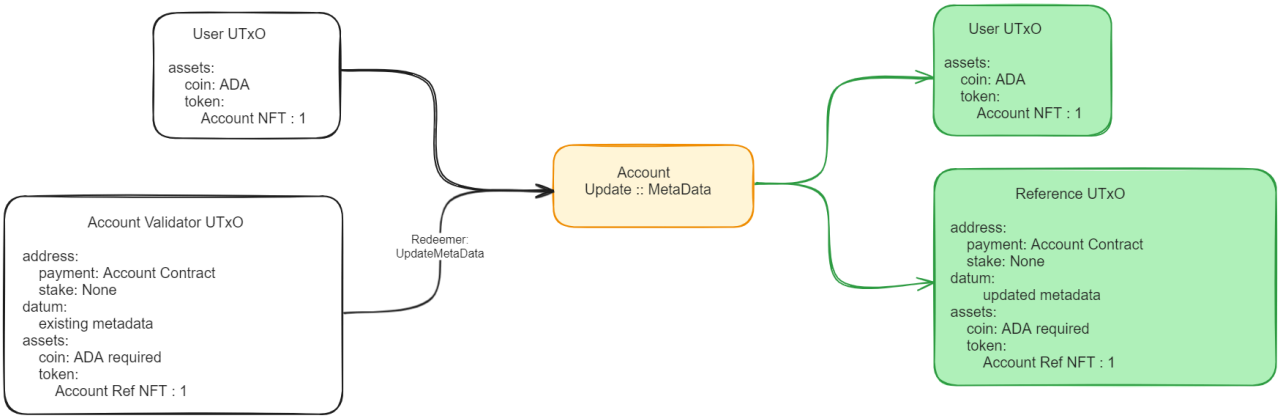

Figure 7: Update Account Metadata UTxO diagram

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 3.3.3.3.1.
   - 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 3.3.3.3.1
   - 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.

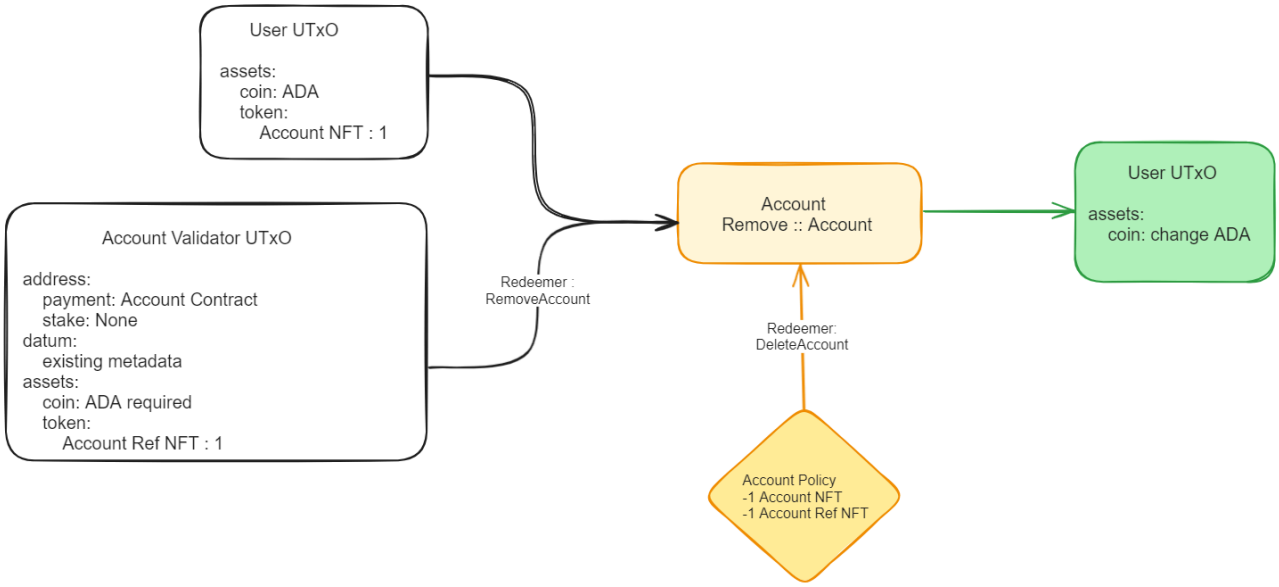

Figure 8: Remove Account Metadata UTxO diagram

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 3.3.3.3.1
- Value:
  - Minimum ADA

- 1 Reference NFT Asset

4.2.4.2. Mints

1. **Account 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 Validator

4.3.1. Mint :: InitiateSubscription

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

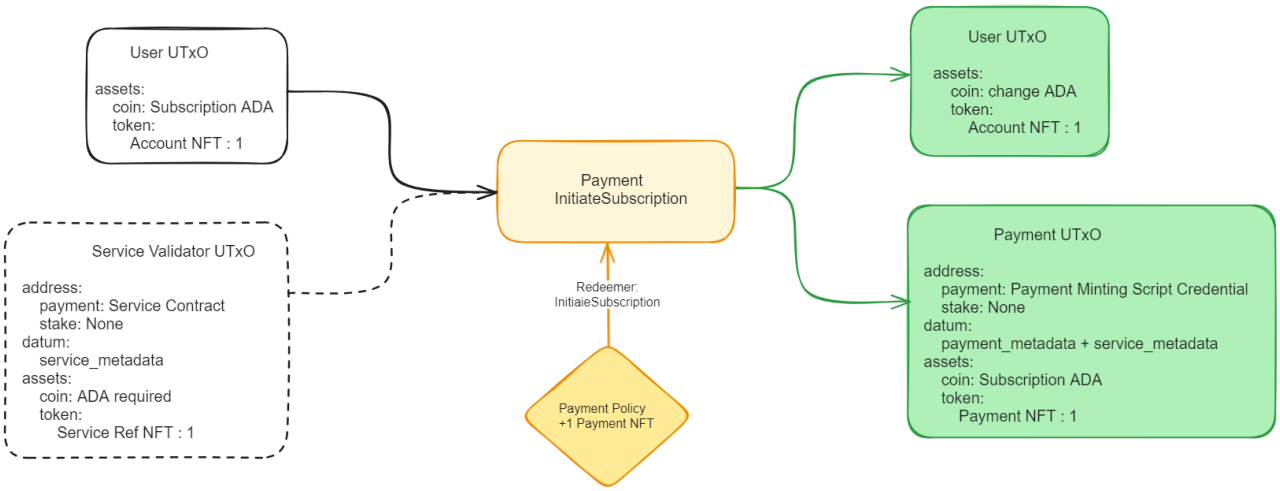

Figure 9: Initiate Subscription UTxO diagram

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

4.3.1.2. Mints

1. **Payment 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.

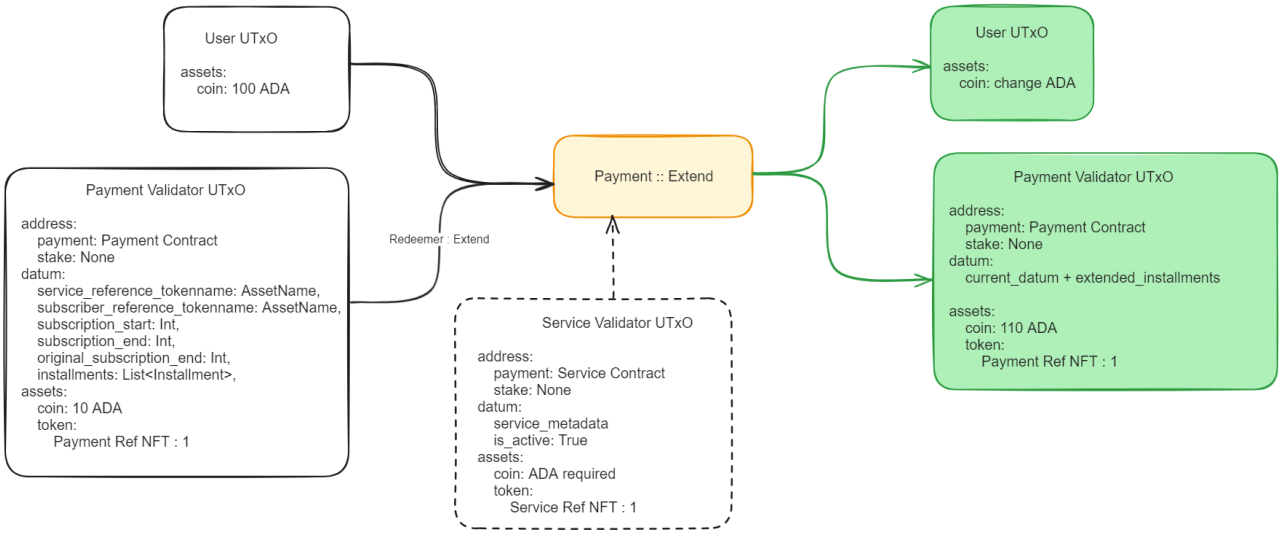

Figure 10: Extend Plan UTxO diagram

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
3. **Service Reference UTxO**
   - Address: Service Contract Address

- Datum:
  - service_datum: listed in Section 3.3.2.3.1
- Value:
  - 1 Service NFT Asset
  - Minimum Ada

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.

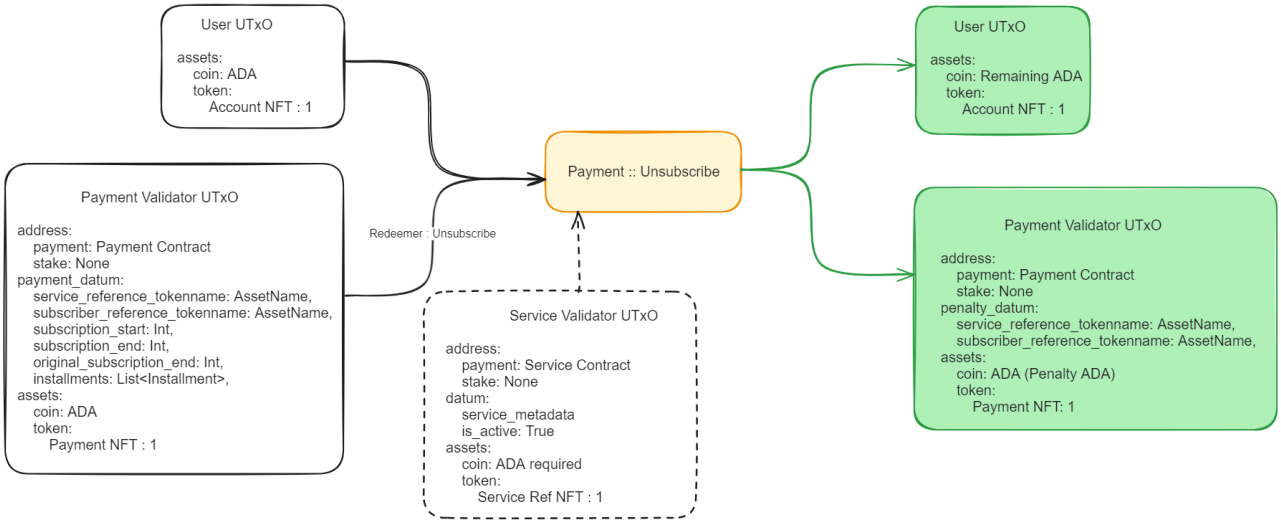

Figure 11: Unsubscribe UTxO diagram

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

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

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 :: Merchant 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.

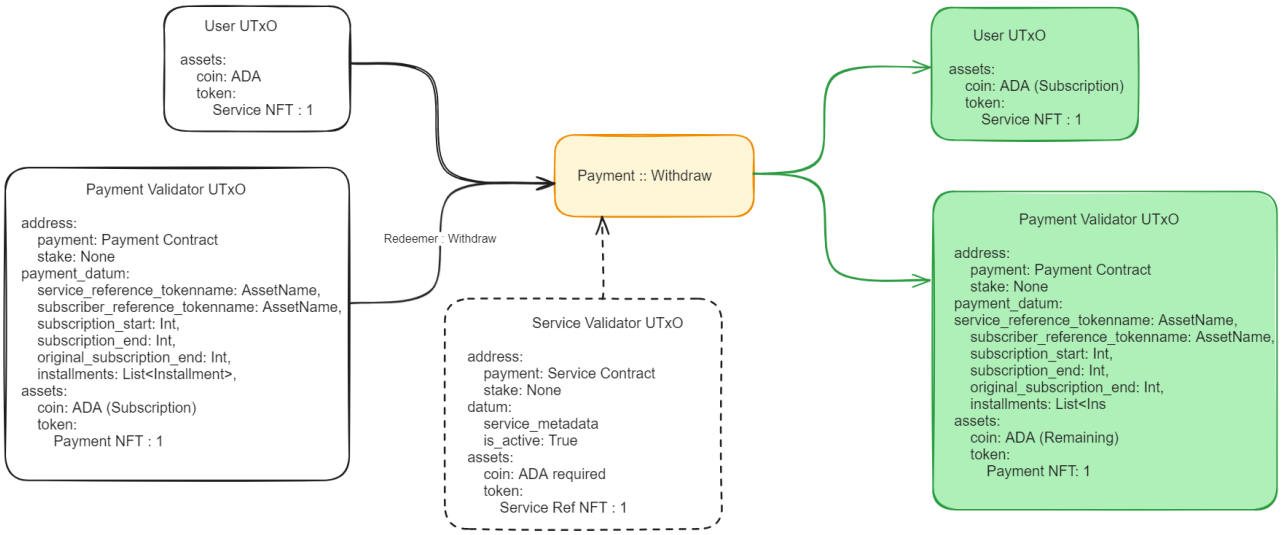

Figure 12: Merchant Withdraw UTxO diagram

4.3.4.1. Inputs:

1. **Merchant Wallet UTxO**
   - Address: Merchant's wallet address
   - Value:
     - Minimum ADA
     - 1 Service NFT Asset
2. **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
3. **Service Reference UTxO**

- Address: Service Contract Address
- Datum:
  - service_datum: listed in Section 3.3.2.3.1
- Value:
  - 1 Service NFT Asset
  - Minimum Ada

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.

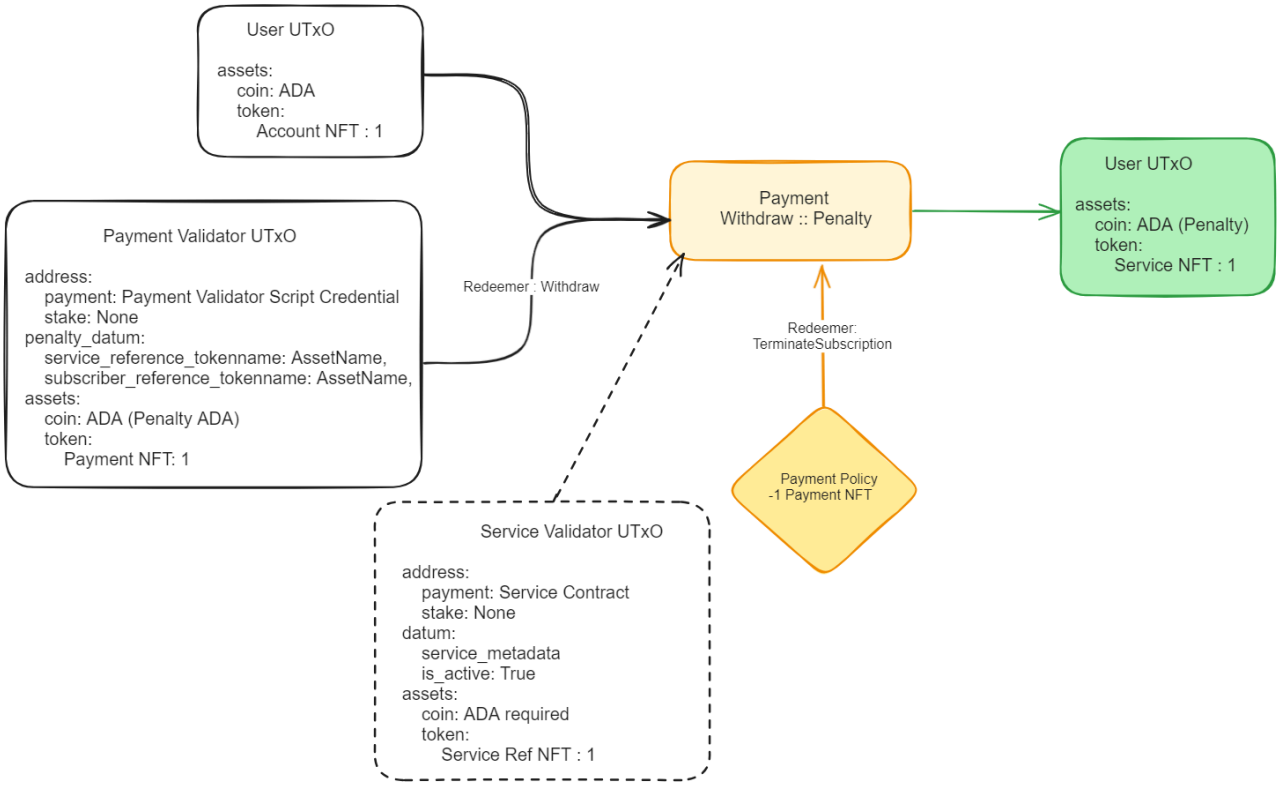

Figure 13: Penalty Withdraw UTxO diagram

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: as Section 3.3.1.4.2

- Value:
  - Minimum ADA
  - Payment NFT Asset

1. **Service Reference UTxO**
   - Address: Service Contract Address
   - Datum:
     - service_datum: listed in Section 3.3.2.3.1
   - Value:
     - 1 Service NFT Asset
     - Minimum Ada

4.3.5.2. Mints

1. **Payment 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.

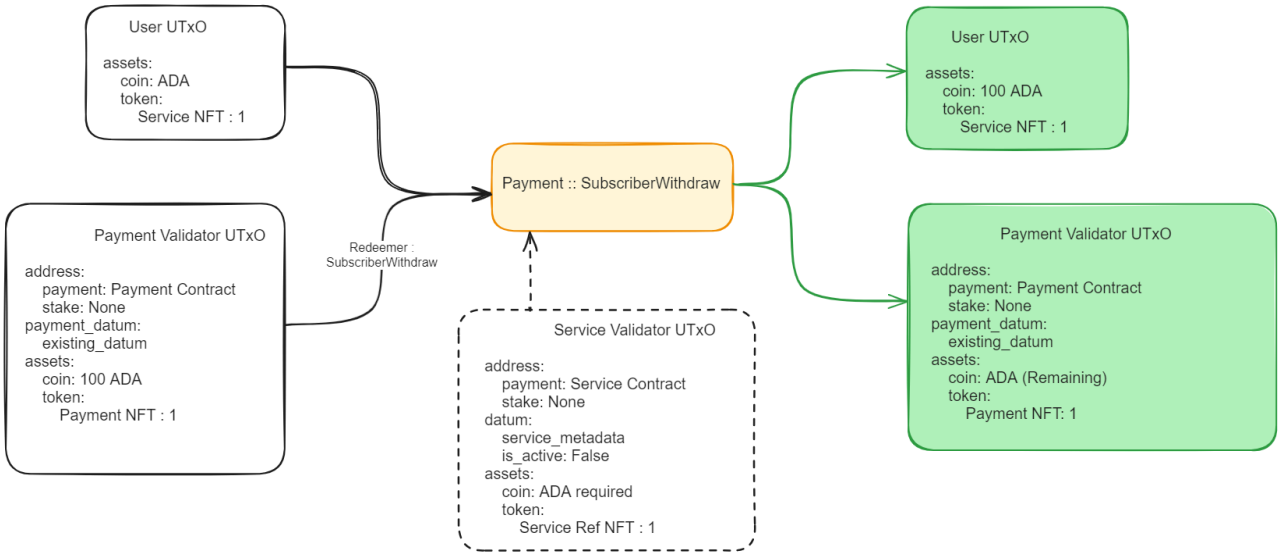

Figure 14: Subscriber Withdraw UTxO diagram

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

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

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