Constant Product DEX Specification

Tomasz Maciosowski
MLabs
tomasz@mlabs.city

Abstract

This document describes a simple DEX validator designed for the Cardano blockchain. This validator allows users directly interact with the liquidity pool, and swap tokens at a predetermined fixed price in any Cardano native tokens. The implementation of this validator will be carried out across multiple programming languages, compiled into UPLC, and tested with a uniform test suite. While the validator is engineered with security in mind, certain functionalities necessary for a production environment may be absent.

1 Introduction

The validator implements a constant product liquidity pool, where users can swap tokens at a predetermined fixed price interacting with the script-owned UTxO directly, rather than through a batching/escrow contract. Users can also deposit and withdraw liquidity from the pool, by providing the necessary pair of tokens to the script-owned UTxO, and receive a proportional amount of the pool's liquidity tokens (LP tokens) in return. LP tokens are specific to each pool, and are used to track the amount of liquidity a user has deposited into the pool. By burning LP tokens, users can withdraw liquidity from the pool, and receive a proportional amount of the tokens from the pool at the current exchange rate. The design does not address concurrency issues, and is not suitable for a production environment, without utilizing tx chaining or batching which are out of scope for this specification.

2 Pool Validator

DEX is a constant product, meaning that after every token swap, the product of the amount of each token in the pool remains constant. Users can interact with the Pool Validator by swapping tokens, depositing liquidity, and withdrawing liquidity. The Pool Validator is parametrized by the following data types:

2.1 Data Types

```
data Redeemer
  = Swap
  | DepositLiquidity
  | WithdrawLiquidity

data Datum = Datum
  { tokenA :: AssetClass
  , tokenB :: AssetClass
  , poolNft :: AssetClass
```

```
  , lpToken :: CurrencySymbol
  , mintedLpTokens :: Integer
  , swapFee :: Integer
  }
```

2.2 Validation Checks

Case 1: Swap

- Output datum remains unchanged.
- The product of the amount of tokens in the pool scaled down by fee did not change.
- No LP tokens are minted.
- Pool NFT is returned.

Case 2: Deposit Liquidity

- Square of minted LP tokens is less than or equal to the product of the tokens deposited.
- Output datum contains the new `mintedLpTokens` value with minted LP amount added.
- Pool NFT is returned.

Case 3: Withdraw Liquidity

- Output datum contains the new `mintedLpTokens` value with burned LP amount subtracted
- The product of the amount of tokens in the pool equals the square of new `mintedLpTokens` value.
- Pool NFT is returned.

3 Pool NFT Minting Policy

Pool NFT is a state token that validates that the liquid pool UTxO holding it has been properly initialized. Pool NFT Minting Policy has one redeemer used to bootstrap a new liquidity pool. The redeemer takes the initial spent UTxO as an argument that needs to be spent in order to create a new pool. Token name of the NFT is derived from the initial spent UTxO to ensure that only one NFT of each name will be ever created.

3.1 Data Types

```
data Redeemer = CreatePool TxOutRef
```

3.2 Validation Checks

- Initial spent UTxO is spent.
- Only one NFT is minted with name set to initial spent.

4 LP Minting Policy

LP tokens represent ownership of a share of the pool. LP tokens are minted when liquidity is deposited into the pool, and burned when liquidity is withdrawn from the pool. LP tokens are specific to each pool, and are used to track the amount of liquidity a user has deposited into the pool. All minting and burning checks are being performed by Pool Validator, so LP Minting Policy forwards all checks to it by making sure that they are being invoked. Minting policy is parametrized by the symbol of Pool NFT.

4.1 Data Types

```
data Redeemer = ForwardCheck
```

4.2 Validation Checks

- Pool NFT is spent.