Table of Contents

Sundaeswap.ts

Introduction:

Sundaeswap is a decentralized exchange (DEX) operating on the Cardano blockchain. It provides liquidity pools for users to trade tokens efficiently. This document provides an overview of the Sundaeswap class implementation, focusing on trading functionalities, liquidity management, and price querying.

Class Definition:

The `Sundaeswap` class is responsible for managing trade execution, estimating trade values, and liquidity positions. It follows a singleton pattern to ensure a single instance per network.

Initialization and Instance Management:

The class maintains instances using a static property:

```
private static _instances: { [name: string]: Sundaeswap };
```

The `getInstance` method returns an instance for the specified network, creating one if necessary.

Trade Estimation:

The class provides methods to estimate the trade value for buying and selling tokens.

- `estimateBuyTrade`: Calculates the required ADA for a given base token amount.
- `estimateSellTrade`: Computes the ADA received for selling a base token.

Both methods use liquidity pool data to perform Automated Market Maker (AMM) calculations with BigNumber for precision.

Executing Trades:

Trades are executed through the executeTrade method, which:

1. Fetches liquidity pool data.
2. Determines the amount needed.
3. Uses TxBuilderLucidV3 to create and submit a transaction.

```
const txBuilder = new TxBuilderLucidV3(lucidInstance, new DatumBuilderLucidV3(network));
const result = await txBuilder.swap({ ...args });
```

Liquidity Management:

The class provides methods for managing liquidity:

- addPosition: Adds liquidity to a pool.
- reducePosition: Reduces liquidity from a pool.

These methods handle token validation and amount conversion using BigNumber for precision.

Fetching Pool Data:

The getPoolData method retrieves pool information from a query provider:

```
async getPoolData(ident: string, queryProvider: any) {
  try {
    const result = await queryProvider.findPoolData({ ident });
    if (!result || !result.liquidity) throw new Error('Invalid pool data');
    return result;
  } catch (error) {
    console.error('Error fetching pool data:', error);
    throw new Error('Failed to fetch pool data');
  }
}
```

This ensures accurate pricing and liquidity information.

Sundaeswap.controller.ts

Introduction:

The sundaeswap.controller.ts file serves as the primary interface for handling trade operations, liquidity management, and price querying in the Sundaeswap decentralized exchange (DEX) on the Cardano blockchain. It integrates with the Sundaeswap class and Cardano blockchain utilities to facilitate transactions.

Imports and Dependencies:

The controller imports several key modules, including:

- Cardano and Sundaeswap for blockchain and DEX interactions.
- Transaction types for handling trade transactions.

- BlockFrostAPI, Lucid, and QueryProviderSundaeSwap for querying blockchain data.
- HttpException for error handling.

Trade Operations:

The controller provides methods to handle trading, price estimation, and liquidity operations. These functions interact with Sundaeswap and Cardano to execute trades securely.

Fetching Trade Information The getTradeInfo function retrieves trade details, including expected amounts and token addresses. It determines if the trade is a buy or sell order and fetches the appropriate trade estimation:

```
async function getTradeInfo(
  cardanoish: Cardano,
  sundaeswapish: Sundaeswap,
  baseAsset: string,
  quoteAsset: string,
  baseAmount: string,
  tradeSide: string,
  network: TSupportedNetworks
): Promise<TradeInfo> {
  const baseToken = cardanoish.getTokenForSymbol(baseAsset);
  const quoteToken = cardanoish.getTokenForSymbol(quoteAsset);
  const queryProvider = new QueryProviderSundaeSwap(network);

  let expectedTrade: ExpectedTrade;
  if (tradeSide === 'BUY') {
    expectedTrade = await sundaeswapish.estimateBuyTrade(...);
  } else {
    expectedTrade = await sundaeswapish.estimateSellTrade(...);
  }

  return { baseToken, quoteToken, expectedTrade };
}
```

Price Estimation:

The price function calculates the trade price by determining the ratio between the expected amount and the base amount. It includes:

- Error handling for failed price estimations.
- Logging request timestamps for latency calculations.

Trade Execution:

The trade function executes a trade by:

1. Fetching wallet details and initializing a Lucid instance.
2. Retrieving token information.
3. Calling executeTrade in the Sundaeswap class.
4. Returning transaction details, including gas costs and execution time.

Liquidity Management:

The controller provides functions to add and remove liquidity:

- addLiquidity: Adds liquidity to the Sundaeswap pool.

- removeLiquidity: Withdraws liquidity from the pool based on a percentage.

Both methods:

- Validate tokens and user balance.
- Retrieve necessary blockchain data.
- Execute transactions using TxBuilderLucidV3.

Utility Functions

- latency(): Calculates the latency between request initiation and completion.
- getBackendLucidInstance(): Initializes and configures a Lucid instance with the Blockfrost API for Cardano transactions.

Sundaeswap.config.ts

Introduction:

The configuration file defines essential TypeScript interfaces used in the Sundaeswap decentralized exchange (DEX) implementation. These interfaces facilitate structured data management for trade execution and transaction tracking.

Interfaces

ExpectedTrade *Interface*

- Represents the expected outcome of a trade operation.
- Contains:

```
interface ExpectedTrade {
  expectedAmount: string; // Final amount after trade execution
  rawAmount: string; // Raw amount in smallest units (e.g., lovelace for ADA)
}
```

TradeInfo *Interface*

- Stores details about a trade operation, including base and quote token information.
- Fields:

```
interface TradeInfo {
  baseToken: CardanoTokenInfo[]; // Information about the base token
  quoteToken: CardanoTokenInfo[]; // Information about the quote token
  requestAmount: string; // Requested trade amount
  expectedTrade: ExpectedTrade; // Expected trade details
  baseTokenAddress: string; // Address of the base token
  quoteTokenAddress: string; // Address of the quote token
}
```

Transaction *Interface*

- Defines the structure of a transaction record in Sundaeswap.
- Fields:

```
interface Transaction {
  hash: string; // Unique transaction hash
  to?: string; // Optional recipient address
```

```
  from?: string; // Optional sender address
  rawAmount: BigInt; // Transaction amount in raw units
}
```