Table of Contents

network.routes.test.ts

Overview

This document outlines the integration tests for the network routes, focusing on status retrieval, configuration, and token retrieval functionalities for the Cardano implementation.

Initialization

Setup and Teardown

- beforeAll: Initializes the Cardano instance in the 'preprod' network before all tests.
- beforeEach: Patches the Cardano chain method before each test.
- afterEach: Unpatches any mocked functions after each test to ensure a clean state.
- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Status Retrieval Tests

Cardano Preprod Network Status

- Purpose: Ensures that the API returns the correct status for the Cardano preprod network.
- Process:
    - Mocks the getCurrentBlockNumber method to return a predefined block number.
    - Sends a GET request to the /chain/status endpoint with the Cardano preprod network parameters.
    - Asserts that the response contains the expected network details, including the chain name and current block number.

Missing Network Parameters

- Purpose: Ensures that the API returns an error when network parameters are not specified.
- Process:
    - Mocks the getCurrentBlockNumber method to return a predefined block number.
    - Sends a GET request to the /chain/status endpoint without specifying network parameters.
    - Asserts that the response status is 503.

Invalid Network

- Purpose: Ensures that the API returns an error for an invalid network.
- Process:

- Sends a GET request to the /chain/status endpoint with invalid network parameters.
- Asserts that the response status is 500.

Configuration Retrieval Test

Valid Request

- Purpose: Ensures that the API returns the configuration details.
- Process:
    - Sends a GET request to the /chain/config endpoint.
    - Asserts that the response status is 200.

Token Retrieval Tests

Cardano Preprod Tokens (No Symbols)

- Purpose: Ensures that the API returns the tokens for the Cardano preprod network when no token symbols are provided.
- Process:
    - Sends a GET request to the /chain/tokens endpoint with the Cardano preprod network parameters.
    - Asserts that the response status is 200.

Cardano Preprod Tokens (With Symbols)

- Purpose: Ensures that the API returns the tokens for the Cardano preprod network when specific token symbols are provided.
- Process:
    - Sends a GET request to the /chain/tokens endpoint with the Cardano preprod network parameters and specific token symbols.
    - Asserts that the response status is 200.

Invalid Chain

- Purpose: Ensures that the API returns an error when an invalid chain is specified.
- Process:
    - Sends a GET request to the /chain/tokens endpoint with invalid chain parameters.
    - Asserts that the response status is 503

wallet.controllers.test.ts

Overview

This document outlines the integration tests for the wallet controller, focusing on adding, retrieving, and removing wallets, including Cardano-related test cases.

Initialization

Setup and Teardown

- beforeAll: Initializes the Cardano instance in the 'preprod' network before all tests and patches the passphrase reading method.
- beforeEach: Patches the passphrase reading method before each test.
- afterEach: Unpatches any mocked functions after each test to ensure a clean state.

- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Add and Retrieve Wallets

Add Cardano Wallet

- Purpose: Ensures that a Cardano wallet can be added successfully.
- Process:
    - Mocks the getWallet method to return a predefined Cardano address.
    - Mocks the encrypt method to return a predefined encrypted private key.
    - Calls the addWallet function with the Cardano private key, chain, and network.
    - Calls the getWallets function to retrieve the list of wallets.
    - Asserts that the Cardano address is present in the retrieved wallets.

Fail to Add Wallet to Unknown Chain

- Purpose: Ensures that adding a wallet to an unknown chain results in an error.
- Process:
    - Calls the addWallet function with an unknown chain and network.
    - Asserts that the function throws an HttpException with the appropriate error message and code.

Add and Remove Wallets

Remove Cardano Wallet

- Purpose: Ensures that a Cardano wallet can be removed successfully.
- Process:
    - Mocks the getWallet method to return a predefined Cardano address.
    - Mocks the encrypt method to return a predefined encrypted private key.
    - Calls the addWallet function with the Cardano private key, chain, and network.
    - Calls the removeWallet function with the Cardano chain and address.
    - Calls the getWallets function to retrieve the list of wallets.
    - Asserts that the Cardano address is not present in the retrieved wallets.

wallet.routes.test.ts

Overview

This document outlines the integration tests for the wallet routes, focusing on adding, retrieving, and removing wallets.

Initialization

Setup and Teardown

- beforeAll: Initializes the Cardano instance in the 'preprod' network before all tests and patches the passphrase reading method.
- beforeEach: Patches the passphrase reading method before each test.

- afterEach: Unpatches any mocked functions after each test to ensure a clean state.
- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Add Wallet

Well-formed Cardano Request

- Purpose: Ensures that a well-formed request to add a Cardano wallet returns a successful response.
- Process:
    - Mocks the getWalletFromPrivateKey method to return a predefined Cardano address.
    - Mocks the encrypt method to return a predefined encrypted private key.
    - Sends a POST request to the /wallet/add endpoint with the Cardano private key, chain, and network.
    - Asserts that the response status is 200.

Remove Wallet

Well-formed Cardano Request

- Purpose: Ensures that a well-formed request to remove a Cardano wallet returns a successful response.
- Process:
    - Mocks the getWalletFromPrivateKey method to return a predefined Cardano address.
    - Mocks the encrypt method to return a predefined encrypted private key.
    - Sends a POST request to the /wallet/add endpoint to add the Cardano wallet.
    - Sends a DELETE request to the /wallet/remove endpoint with the Cardano address and chain.
    - Asserts that the response status is 200.

Retrieve Wallets

Well-formed Cardano Request

- Purpose: Ensures that a well-formed request to retrieve wallets returns a successful response.
- Process:
    - Mocks the getWalletFromPrivateKey method to return a predefined Cardano address.
    - Mocks the encrypt method to return a predefined encrypted private key.
    - Sends a POST request to the /wallet/add endpoint to add the Cardano wallet.
    - Sends a GET request to the /wallet endpoint.
    - Asserts that the response status is 200 and that the Cardano address is present in the retrieved wallets.

wallet.validators.test.ts

Overview

This document outlines the unit tests for the wallet validators, focusing on validating Cardano private keys, chain, and addresses.

Test Cases

Cardano Private Key Validation

Well-formed Private Key

- Purpose: Ensures that a well-formed Cardano private key passes validation.
- Process:
    - Calls the isCardanoPrivateKey function with a valid Cardano private key.
    - Asserts that the function returns true.

Invalid Private Key

- Purpose: Ensures that an invalid string fails validation as a Cardano private key.
- Process:
    - Calls the isCardanoPrivateKey function with an invalid string.
    - Asserts that the function returns false.

Private Key Validation

Valid Cardano Private Key

- Purpose: Ensures that a request with a valid Cardano private key passes validation.
- Process:
    - Calls the validatePrivateKey function with a request containing a valid Cardano private key and chain.
    - Asserts that the function returns an empty array, indicating no errors.

Chain Validation

Valid Cardano Chain

- Purpose: Ensures that a request with the Cardano chain passes validation.
-
- Process:
    - Calls the validateChain function with a request containing the Cardano chain.
    - Asserts that the function returns an empty array, indicating no errors.

Address Validation

Valid Cardano Address

- Purpose: Ensures that a request with a valid Cardano address passes validation.
- Process:
    - Calls the validateAddress function with a request containing a valid Cardano address.
    - Asserts that the function returns an empty array, indicating no errors.

cardano.controllers.test.ts

Overview:

This document outlines the integration tests for the Cardano blockchain implementation, focusing on initialization, token retrieval, and balance checking functionalities.

Initialization

Setup and Teardown

- beforeAll: Initializes the Cardano instance in the 'preprod' network before all tests.
- afterEach: Unpatches any mocked functions after each test to ensure a clean state.
- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Initialization Test

- Purpose: Ensures that the init method waits for the first call to complete before allowing subsequent calls.
- Process:
    - Calls init and waits for it to complete.
    - Calls init again and checks that the first call was completed before the second call proceeds.

Token Retrieval Test

- Purpose: Verifies that the correct token information is returned for a given symbol.
- Process:
    - Mocks the getTokenForSymbol method to return a predefined token.
    - Asserts that the method returns the correct token information when queried with the symbol 'MIN'.

Balance Check Test

- Purpose: Checks that an error is thrown if the wallet is not found.
- Process:
    - Mocks the getWalletFromAddress method to throw an error indicating the wallet does not exist.
    - Calls the balances method of CardanoController and asserts that it throws the expected error.

cardano.routes.test.ts

Overview

This document outlines the integration tests for the Cardano blockchain API, focusing on balance retrieval and transaction polling functionalities.

Initialization

Setup and Teardown

- beforeAll: Initializes the Cardano instance in the 'preprod' network before all tests.
- afterEach: Unpatches any mocked functions after each test to ensure a clean state.
- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Balance Retrieval Tests

Supported Tokens

- Purpose: Ensures that the API returns the correct balances for supported tokens.
- Process:
    - Mocks the necessary methods to return predefined wallet, token, and balance information.
    - Sends a POST request to the /chain/balances endpoint with supported token symbols ('ADA', 'MIN').
    - Asserts that the response contains balances for the specified tokens.

Native Token

- Purpose: Ensures that the API returns the correct balance for the native token.
- Process:
    - Mocks the necessary methods to return predefined wallet, token, and balance information.
    - Sends a POST request to the /chain/balances endpoint with the native token symbol ('ADA').
    - Asserts that the response contains the balance for the native token.

Unsupported Tokens

- Purpose: Ensures that the API returns an error for unsupported tokens.
- Process:
    - Mocks the necessary methods to return predefined wallet, token, and balance information.
    - Sends a POST request to the /chain/balances endpoint with unsupported token symbols ('XXX', 'YYY').
    - Asserts that the response status is 500.

Invalid Parameters

- Purpose: Ensures that the API returns a 404 error when required parameters are missing.
- Process:
    - Sends a POST request to the /chain/balances endpoint without the required token symbols.
    - Asserts that the response status is 404.

Transaction Polling Tests

Network Unavailable

- Purpose: Ensures that the API returns a network error when the network is unavailable.
- Process:
    - Sends a POST request to the /chain/poll endpoint with a valid transaction hash but an unavailable network.
    - Asserts that the response status is 503 and contains the appropriate error code and message.

Unknown Error

- Purpose: Ensures that the API returns an unknown error when an unexpected error occurs.
- Process:
    - Mocks the getTransaction method to throw an error.
    - Sends a POST request to the /chain/poll endpoint with a valid transaction hash.
    - Asserts that the response status is 503 and contains the appropriate error code.

Successful Query

- Purpose: Ensures that the API returns the correct status for a successful transaction query.
- Process:
    - Mocks the getTransaction method to return a successful transaction.
    - Sends a POST request to the /chain/poll endpoint with a valid transaction hash.
    - Asserts that the response status is 200 and contains the transaction details.

Invalid Transaction Hash

- Purpose: Ensures that the API returns an unknown error for an invalid transaction hash.
- Process:
    - Sends a POST request to the /chain/poll endpoint with an invalid transaction hash.
    - Asserts that the response status is 503 and contains the appropriate error code and message.

cardano.validators.test.ts

Overview

This document outlines the unit tests for the Cardano address and asset symbol validators, ensuring they correctly validate input data.

Test Cases

Cardano Address Validation

Well-formed Address

- Purpose: Ensures that a well-formed Cardano address passes validation.
- Process:
    - Provides a valid Cardano address starting with 'addr_test'.
    - Asserts that the validation function returns an empty array, indicating no errors.

Invalid Prefix

- Purpose: Ensures that an address with an invalid prefix fails validation.
- Process:
    - Provides an address that does not start with 'addr' or 'addr_test'.
    - Asserts that the validation function returns an error message indicating an invalid Cardano address.

Non-hexadecimal Characters

- Purpose: Ensures that an address with non-hexadecimal characters fails validation.
- Process:

- Provides an address containing non-hexadecimal characters.
- Asserts that the validation function returns an error message indicating an invalid Cardano address.

Asset Symbols Validation

Well-formed Symbols

- Purpose: Ensures that well-formed asset symbols pass validation.
- Process:
  - Provides a list of valid asset symbols ('ADA', 'MIN').
  - Asserts that the validation function returns an empty array, indicating no errors.

Empty Symbol

- Purpose: Ensures that an empty asset symbol fails validation.
- Process:
  - Provides a list containing an empty string as an asset symbol.
  - Asserts that the validation function returns an empty array, indicating no errors.

minswap.routes.test.ts

Overview

This document outlines the integration tests for the Minswap routes, focusing on price retrieval and trade execution functionalities.

Initialization

Setup and Teardown

- beforeAll: Initializes the Express app, Cardano instance, and Minswap instance in the 'preprod' network before all tests.
- afterEach: Unpatches any mocked functions after each test to ensure a clean state.
- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Price Retrieval Tests

Buy Price

- Purpose: Ensures that the API returns the correct price for a buy order.
- Process:
  - Mocks the necessary methods to return predefined wallet, token, and trade information.
  - Sends a POST request to the /amm/price endpoint with a buy order.
  - Asserts that the response contains the expected amount and raw amount.

Sell Price

- Purpose: Ensures that the API returns the correct price for a sell order.
- Process:
  - Mocks the necessary methods to return predefined wallet, token, and trade information.
  - Sends a POST request to the /amm/price endpoint with a sell order.

- Asserts that the response contains the expected amount and raw amount.

Unrecognized Quote Symbol

- Purpose: Ensures that the API returns an error for an unrecognized quote symbol.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/price endpoint with an unrecognized quote symbol.
    - Asserts that the response status is 500.

Unrecognized Base Symbol

- Purpose: Ensures that the API returns an error for an unrecognized base symbol.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/price endpoint with an unrecognized base symbol.
    - Asserts that the response status is 500.

Trade Execution Tests

Buy Trade

- Purpose: Ensures that the API executes a buy trade correctly.
- Process:
    - Mocks the necessary methods to return predefined wallet, token, and trade information.
    - Sends a POST request to the /amm/trade endpoint with a buy order.
    - Asserts that the response contains the transaction hash.

Sell Trade

- Purpose: Ensures that the API executes a sell trade correctly.
- Process:
    - Mocks the necessary methods to return predefined wallet, token, and trade information.
    - Sends a POST request to the /amm/trade endpoint with a sell order.
    - Asserts that the response contains the transaction hash.

Incorrect Parameters

- Purpose: Ensures that the API returns a 404 error when required parameters are incorrect.
- Process:
    - Sends a POST request to the /amm/trade endpoint with incorrect parameters.
    - Asserts that the response status is 404.

SwapExactInTx Operation Failure

- Purpose: Ensures that the API returns an error when the swapExactInTx operation fails.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Mocks the swapExactInTx method to return an error.
    - Sends a POST request to the /amm/trade endpoint with a sell order.
    - Asserts that the response status is 500.

SwapExactOutTx Operation Failure

- Purpose: Ensures that the API returns an error when the swapExactOutTx operation fails.

- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Mocks the swapExactOutTx method to return an error.
    - Sends a POST request to the /amm/trade endpoint with a sell order.
    - Asserts that the response status is 500.

minswap.lp.routes.test.ts

Overview

This document outlines the integration tests for the Minswap liquidity routes, focusing on price retrieval, liquidity addition, and liquidity removal functionalities.

Initialization

Setup and Teardown

- beforeAll: Initializes the Express app, Cardano instance, and Minswap instance in the 'preprod' network before all tests.
- afterEach: Unpatches any mocked functions after each test to ensure a clean state.
- afterAll: Closes the Cardano instance after all tests to release resources.

Test Cases

Price Retrieval Tests

Valid Parameters

- Purpose: Ensures that the API returns the correct price when all parameters are valid.
- Process:
    - Mocks the necessary methods to return predefined token and price information.
    - Sends a POST request to the /amm/liquidity/price endpoint with valid parameters.
    - Asserts that the response status is 200.

Invalid Fee

- Purpose: Ensures that the API returns an error when the fee parameter is invalid.
- Process:
    - Mocks the necessary methods to return predefined token information.
    - Sends a POST request to the /amm/liquidity/price endpoint with an invalid fee parameter.
    - Asserts that the response status is 404.

Liquidity Addition Tests

Valid Parameters

- Purpose: Ensures that the API adds liquidity correctly when all parameters are valid.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/liquidity/add endpoint with valid parameters.
    - Asserts that the response status is 200 and contains the transaction hash.

Unrecognized Token Symbol

- Purpose: Ensures that the API returns an error for an unrecognized token symbol.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/liquidity/add endpoint with an unrecognized token symbol.
    - Asserts that the response status is 500.

Invalid Fee Tier

- Purpose: Ensures that the API returns an error for an invalid fee tier.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/liquidity/add endpoint with an invalid fee tier.
    - Asserts that the response status is 404.

Liquidity Removal Tests

Valid Parameters

- Purpose: Ensures that the API removes liquidity correctly when all parameters are valid.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/liquidity/remove endpoint with valid parameters.
    - Asserts that the response status is 200 and contains the transaction hash.

Invalid Token ID

- Purpose: Ensures that the API returns an error for an invalid token ID.
- Process:
    - Mocks the necessary methods to return predefined wallet and token information.
    - Sends a POST request to the /amm/liquidity/remove endpoint with an invalid token ID.
    - Asserts that the response status is 404.