Table of Contents

[Application Flow Chart 1](#_Toc167019333)

[How to Start Application 2](#_Toc167019334)

[Open Account API 2](#_Toc167019335)

[Query Entity API 4](#_Toc167019336)

[Update Account Status API 6](#_Toc167019337)

[Fund In API 8](#_Toc167019338)

[Fund Out API 11](#_Toc167019339)

[Move Asset API 14](#_Toc167019340)

[Query Account API 18](#_Toc167019341)

[Query Wallet API 19](#_Toc167019342)

[Query Movement API 20](#_Toc167019343)

[Query historical balance of wallet API 22](#_Toc167019344)

## Application Flow Chart

A diagram of a projector

Description automatically generated

The project is following the Command Query Responsibility Segregation (CQRS) and Event Sourcing design patterns. In-memory storage (ConcurrentHashMap) is used.  
When the application receives request, it will validate the command, handle the command (in aggregator) and write events into event-store and the event has a temporal order by entity id in ConcurrentLinkedQueue.

There is a thread which is checking any pending entity id in the ConcurrentLinkedQueue. If yes, it will process events for this entity in the event-store and update the result in read repository.

The application will query the read repository for entity, account and wallet information, query the event-store for the historical balance and asset movements.

As we are using the in-memory storage and there is no database, we cannot use transactional management in spring boot and the rollback function for database, we cannot fully meet the request of “execute all or nothing”, instead we are trying to do full validation (request body validation + business logic validation) for the whole request before processing.

Details of request body validation and business logic validation for each API will be listed in the document.

## How to Start Application

* Go to the project directory
* Run Command to build the application: mvn package



A screenshot of a computer program

Description automatically generated

* Run command to start the application: mvn spring-boot:run



## Open Account API

Create entity and its accounts/wallets. The creation is processed in async mode.

POST <http://localhost:8080/api/openAccount>

Sample Request Body:

{

"entityName" : "HenryEntity",

"account" : [

{"accountName":"Low risk account",

"wallets": [

{"walletName": "Wallet BOND 01", "type": "BOND"},

{"walletName": "Wallet BOND 02", "type": "BOND"}

]

},

{"accountName":"High risk account 02",

"wallets": [

{"walletName": "Wallet STOCK 01", "type": "STOCK"},

{"walletName": "Wallet STOCK 02", "type": "STOCK"},

{"walletName": "Wallet CRYPTO", "type": "CRYPTO"}

]

}

]

}

The API will return the entityId, details refer to the Postman captured screen.

A screenshot of a computer

Description automatically generated

Request body validation

* entityName is mandatory
* account is mandatory
* accountName is mandatory
* walletName is mandatory
* wallet type shall be either BOND/STOCK/CRYPTO/CURRENCY

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Query Entity API

Get entity information and its accounts/wallets by entityId

GET [http://localhost:8080/api/query/entity/{entityId}](http://localhost:8080/api/query/entity/%7bentityId%7d)

For this example, the entityId is coming from the response of the “Open Account API”. In the response of “Query Entity API”, we can get all accounts / wallets in the entity.

A screenshot of a computer

Description automatically generated

Request body validation-

* The {entityId} passed in the GET request must be a UUID

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

Business Logic Validation-

If the client is passing incorrect entity id (for example, the entity id is UUID but doesn’t exist), it will return error message –

Client made a bad request - Incorrect entityId provided.

A screenshot of a computer

Description automatically generated

## Update Account Status API

The account has two status – OPEN and CLOSED. This API can update account status from one to another. The status update is processed in async mode.

POST <http://localhost:8080/api/updateAccountStatus>

Sample Request Body  
{

"id" : "ed3e90f2-8cfd-4740-9c6d-2cc79ff3de4b",

"status": "CLOSED"

}  
A screenshot of a computer

Description automatically generated

Request Body Validation

* id is mandatory
* id must be a UUID
* status is mandatory
* status shall be either OPEN or CLOSED

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Business Logic Validation

If the client is trying to update an account which doesn’t exist, API will return the error message –

Client made a bad request - Incorrect accountId {accountId} provided.

A screenshot of a computer

Description automatically generated

If the account is OPEN (CLOSED) and the client still trying to update it to OPEN (CLOSED). API will return the error message –

Client made a bad request - Update account status declined, need to update status from one to another.

A screenshot of a computer

Description automatically generated

## Fund In API

Move in assets to one wallet, multiple move-ins can be processed in a single request.

The movement is processed in async mode.

POST <http://localhost:8080/api/fundIn>

Sample Request Body:

{ "requests":

    [

        {"walletId": "5dd5e863-17ee-4daa-bd84-2f1a0fd3a98d", "amount": 30.0},

        {"walletId": "1a08db2f-c87a-432e-8b08-374426a05ac6", "amount": 20.0}

    ]

}

A screenshot of a computer

Description automatically generated

Request Body Validation

* requests is mandatory
* walletId is mandatory
* walletId must be a UUID
* amount is mandatory and must be positive

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Business Logic Validation

If the client requests move-ins for incorrect walletId (for example, the wallet doesn’t exist), the API will return error message –  
Client made a bad request - Incorrect walletId {walletId} provided.

A screenshot of a computer

Description automatically generated

If the client tries to move assets to wallet in a CLOSED account, the API will return error message –

Client made a bad request - The account {accountId} is CLOSED, we cannot do movement for wallet {walletId} in CLOSED account.

A screenshot of a computer

Description automatically generated

If the client tries to do multiple move-ins to wallets from different entities, the API will return error message –

Client made a bad request - all movements shall be in the same entity.

A screenshot of a computer

Description automatically generated

## Fund Out API

Move out assets from one wallet, multiple move-outs can be processed in a single request

The movement is processed in async mode.

POST <http://localhost:8080/api/fundOut>

Sample Request Body:

{ "requests":

    [

        {"walletId": "fb3453fb-cdcb-47e0-8e68-072552a79ee7", "amount": 30.0},

        {"walletId": "ab7274ed-76b2-46ba-b59f-536ac9f903dc", "amount": 20.0}

    ]

}

A screenshot of a computer

Description automatically generated

Request Body Validation

* requests is mandatory
* walletId is mandatory
* walletId must be a UUID
* amount is mandatory and must be positive

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Business Logic Validation

If the client requests move-outs for incorrect walletId (for example, the wallet doesn’t exist), the API will return error message –  
Client made a bad request - Incorrect walletId {walletId} provided.

A screenshot of a computer

Description automatically generated

If the client tries to move out assets from wallet in a CLOSED account, the API will return error message –

Client made a bad request - The account {accountId} is CLOSED, we cannot do movement for wallet {walletId} in CLOSED account.

A screenshot of a computer

Description automatically generated

If the client tries to do multiple move-outs to wallets from different entities, the API will return error message –

Client made a bad request - all movements shall be in the same entity.

A screenshot of a computer

Description automatically generated

If the wallet doesn’t have sufficient balance for the move-out request, the API will return error message –

Client made a bad request - Insufficient balance for wallet id {walletId}

A screenshot of a computer

Description automatically generated

## Move Asset API

Move assets from one wallet to another, multiple movements can be processed in a single request

The movement is processed in async mode.

POST <http://localhost:8080/api/moveAsset>

Sample Request Body:

{ "requests":

    [

        {"fromWalletId": "8a494c36-df3f-42c0-a120-bf3706c1a7ff", "toWalletId": "82e6dccf-db8e-468a-b349-83fda503e935", "amount": 1.0},

        {"fromWalletId": "8a494c36-df3f-42c0-a120-bf3706c1a7ff", "toWalletId": "a8c3ae09-d2a6-4b23-9218-15b54fdd69bf", "amount": 1.0}

    ]

}

A screenshot of a computer

Description automatically generated

Request Body Validation

* requests is mandatory
* fromWalletId is mandatory
* fromWalletId must be a UUID
* toWalletId is mandatory
* toWalletId must be a UUID
* amount is mandatory and must be positive

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Business Logic Validation

If the client requests movements for incorrect walletId (for example, the wallet doesn’t exist), the API will return error message –  
Client made a bad request - Incorrect walletId {walletId} provided.

A screenshot of a computer

Description automatically generated

If the client tries to move assets from/to wallet in a CLOSED account, the API will return error message –

Client made a bad request - The account {accountId} is CLOSED, we cannot do movement for wallet {walletId} in CLOSED account.

A screenshot of a computer

Description automatically generated

If the client tries to do multiple movements for wallets from different entities, the API will return error message –

Client made a bad request - all movements shall be in the same entity.

A screenshot of a computer

Description automatically generated

If the wallet doesn’t have sufficient balance for the movement request, the API will return error message –

Client made a bad request - insufficient balance for wallet id {walletId}

A screenshot of a computer

Description automatically generated

## Query Account API

Get account information and its wallets by accountId

GET [http://localhost:8080/api/query/account/{accountId}](http://localhost:8080/api/query/account/%7baccountId%7d)

A screenshot of a computer

Description automatically generated

Request Body Validation

* The {accountId} passed in the GET request must be a UUID

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

Business Logic Validation

If the client is passing incorrect account id (for example, the accountId doesn’t exist), it will return error message –

Client made a bad request - Incorrect accountId {accountId} provided.

A screenshot of a computer

Description automatically generated

## Query Wallet API

Get wallet information by walletId  
GET http://localhost:8080/api/query/wallet/{walletId}

A screenshot of a computer

Description automatically generated

Request Body Validation

* The {walletId} passed in the GET request must be a UUID

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

Business Logic Validation

If the client is passing incorrect wallet id (for example, the walletId doesn’t exist), it will return error message –

Client made a bad request - Incorrect walletId {walletId} provided.

A screenshot of a computer

Description automatically generated

## Query Movement API

Query all historical movements of assets within this entity. In the response, the “amount” means the delta (amount to move-in / move-out). The movement is sorted from most recent datetime to the oldest datetime.

GET [http://localhost:8080/api/query/movement/{entityId}](http://localhost:8080/api/query/movement/%7bentityId%7d)

A screenshot of a computer

Description automatically generated

Request Body Validation

* The {entityId} passed in the GET request must be a UUID

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

## Query historical balance of wallet API

Query wallet balance at a given timestamp by walletId. In the request body, the “id” is the walletId, the “datetime” format shall be yyyy-MM-dd HH:mm:ssZ

POST <http://localhost:8080/api/query/walletHistoricalBalance>

Sample Request Body:

{

"id": "6baaaded-9823-40dc-a93b-6aa54d2042a5",

"datetime": "2024-05-15 13:48:00+0000"

}

A screenshot of a computer

Description automatically generated

Request Body Validation

* id is mandatory
* datetime is mandatory
* id must be a UUID
* datetime must be in format of yyyy-MM-dd HH:mm:ssZ. For example, 2024-05-15 13:48:00+0000

Details refer to Postman captured screen below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated