Code2Connect 2023
FICC Tech Problem Statement –Forward Pricing Engine

Contents

1 Background and Introduction

An FX forward is a contractual agreement between the client and the financial institution, to exchange a pair of currencies at a set rate on a future date.

Business that engages in international trade or deal with foreign currency transactions need to manage their currency risk exposure. They engage with a financial institution to provide the forward exchange rate for a future transaction, allowing them to protect themselves against adverse exchange rate movements.

A FX (Foreign Exchange) forwards pricing engine based on trades is used by financial institutions that deal with foreign currency transactions. The pricing engine is responsible for calculating the forward exchange rate of two currencies and providing a quote for a future transaction.

For example, a company might be purchasing raw materials from a foreign supplier, and they have agreed on a price in the foreign currency. However, the actual transaction will take place in the future, and then, the exchange rate might have changed, exposing the company to forex risk. The FX forwards trade allows the company to lock the in the exchange rate at the time of the agreement, protecting them from any potential currency fluctuations.

Similarly financial institutions that deal with foreign exchange transactions use FX forwards pricing engine to manage their risk exposure.

They can calculate their future cash flows or positions accurately, enabling them to hedge their currency risk.

Ultimately having a FX forwards pricing engine based on trades allows businesses and financial institutions to operate more efficiently and reduce their currency risk exposure. They can make more informed decisions about future transactions, manage their cash flow effectively and optimize profitability.

In order to hedge risk a trader should be able to determine following points at various intervals

1. Bid and Ask
2. QuoteStatus – TRADABLE/NON-TRADEABLE/EXCEPTION

Your task for this challenge is to develop a Forward Pricing engine to show above data points at various intervals as well as real time.

2 Bid and Ask Calculations

Bid and Ask can be calculated as per formula given below

Bid = New Mid -(0.5 *Spread /10000)

Ask= New Mid +(0.5*spread) / 10000)

New Mid is calculated as below:

New Mid = rate -Skew

Skew is calculated as below:

Skew = Net position / Skew Ratio Divisor * Variance

Variance is a linear equation, which is define as:

mx+b

x = number of days from trade dates

m = config value (provided)

b = config value (provided)

Refer to provided excel sheet for various examples on Bid and Ask calculation and calculation of number of days.

 You can use excel scenarios to create your test cases.

3 Quote Status determination

It can be of three types:

1. **EXCEPTION** – Unable to calculate due to missing data points.
2. **NON-TRADABLE**
    a. Bid is greater than Ask
    b. Ask or Bid varies over 10% of rate (FXMidEvent)
3. **TRADABLE** – If none of the above conditions are met.

.

4 Requirement for the challenge

For this challenge, you must simulate real world pricing engine. You have to work on following components:

4.1 Essential Components

1. Read data from json file **events.json** and create components /interface listed below. You can add/remove more components as per your design:
    a. **Market Data Producer**: Implement a market data producer that publishes FX prices and config on regular interval. You can induce delay of 1-5 seconds in publishing to simulate real world. This will help to create real-time dashboard to tick when FX event comes in.
    b. **Trade Event Data Producer:** Implement a trade event data producer that publishes trade event on regular interval. You can induce delay of 1-5 seconds in publishing.
    c. **Pricing Engine** -Subscribes to market data and trade event to show real time **Bid and Ask** calculation as well as **QuoteStatus**
    d. Report Generator - Implement a reporting interface to query the **Bid**, **Ask**, **Position** and **QuoteStatus** safter a certain number of event messages have been processed.
    e. **Dashboard -**User Interface to show ticking data.

2. A Microsoft PowerPoint presentation deck that does not exceed 7 pages to display and explain your solutions. A suggested format is in Presentation:

4.1.1 Implementation boundaries

1. Though market data and event data ticks separately they should follow sequential order as per EventId.
2. Report generator queries are inclusive of given EventId.
3. Number of currencies are not limited to initial data set.
4. All final output data should be round up to four decimal places. Your final value could differ few points based on rounding up at various level, but we believe it's possible to match the exact output. Try to get as close as possible to sample output file provided.
5. You are free to use any language and libraries. **Do not use any code generation platform**.

4.1.2 Requirements

1. Provide clear documentation with instructions to run the application and other implementation details.
2. Your program should be well documented in-line and should explain itself.
3. Your program should produce relevant logs as and when needed.
4. We will provide one set of validation input and output files in the format defined in section Inputs and section output This is only for your reference.
5. We will be giving testing input files in the format defined in input section an hour before final submission. We will also mention set of EventIds you need to process. You are required to **process all of them and send back your results** in given output format This is used to evaluate your work.

4.1.3 Assumptions

- We will be using some hypothetical currencies for our data sets.
- All the results which we report from our system will be reported in "USX".
- Sell trades are valid before a buy trade.
- Validation and correction should be done by the code for the dataset.
- Use your best judgement for edge cases and scenarios or reach out to mentors.

4.1.4 Marking Scheme

Item	Points
Overall design, architecture, algorithm, performance, documentation, coding practice, test cases and edge cases.	20
Set up a functioning essential component - **Market Data Producer, Trade Event Data Producer, Pricing Engine.**	20
Set up a functioning essential component-**Report Generator** to generate proper results as per given query and format	20
Presentation deck on thinking process, design considerations, algorithm complexity, solutions and challenges faced.	20
Dashboard -User Interface to show ticking data. Be creative and implement a simple user-friendly design. Can be a CLI as well.	20
Subtotal	100

Table 1: Marking Scheme

5 Sample Input and Output

Refer to sample files provide for input and output .

5.1 Input

5.1.1 events.json

a. Contains three types of events:

i. **TradeEvent** – Denotes buy/sell event of a particular product.

```
16      {
17          "EventId": 3,
18          "EventType": "TradeEvent",
19          "BuySell": "buy",
20          "Ccy": "GBX",
21          "Tenor": "1M",
22          "Quantity": 10000,
23          "TradeId": "T1"
24      },
```

Table 2: TradeEvent from events.json

ii. **FXMidEvent** – Denotes change in current FX for a particular currency.

```
3           "EventId": 1,
4           "EventType": "FXMidEvent",
5           "Ccy": "GBX",
6           "rate": 1.22
7       },
```

Table 3: FXMidEvent from events.json

iii. **ConfigEvent** – Denotes change in config values of linear equation for skew calculation

```
 8      {
 9          "EventId": 2,
10          "EventType": "ConfigEvent",
11          "m": 0.01,
12          "b": 0.08,
13          "DivisorRatio": 100000,
14          "Spread": 2
15      },
```

Table 4: ConfigEvent from events.json

5.1.2 input.json

We will provide you set of EventIds, Ccy, Tenor an hour before submission with a file name input.json. EventIds will be inclusive with random orders and can be repeated. You should run you code against those set of input and provide us output as per instruction given below:

1. It will have a Json record with Ccy, Tenor and EventId . Refer to provided sample input file.

Table 5: Sample record in Input.json file for EventId

5.2 Output

5.2.1 output.json

- For each EventId a row to be added as per below example.

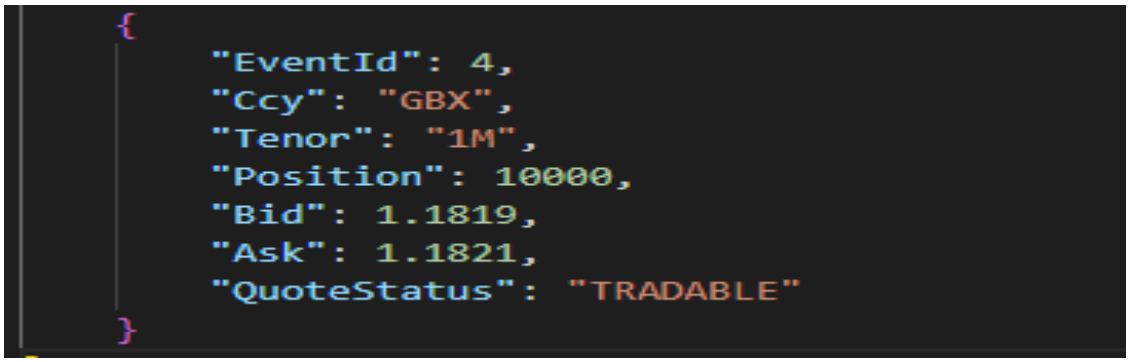

Table 6: Sample record in Output.json file for TRADABLE

- If values cannot be calculated for a particular position due to missing data points, QuoteStatus should be mark as an 'EXCEPTION' and Bid /Ask should be shown as 'NA'. Position will still hold valid.

```
{
    "EventId": 3,
    "Ccy": "GBX",
    "Tenor": "1M",
    "Position": 10000,
    "Bid": "NA",
    "Ask": "NA",
    "QuoteStatus": "EXCEPTION"
}
```


Table 7: Sample record in Output.json file for EXCEPTION

- If position (key -> Ccy and Tenor) is not available till that EventId mark the record as an 'EXCEPTION' and Position as 'NA' along with Bid and Ask.


```
{
    "EventId": 4,
    "Ccy": "GBX",
    "Tenor": "6M",
    "Position": "NA",
    "Bid": "NA",
    "Ask": "NA",
    "QuoteStatus": "EXCEPTION"
}
```


Table8: Sample record in Output.json if no Position exist

5.2.2 Dashboard

There is no specific format. You can create a basic dashboard using command line to show ticking data or come up with nice user interface using web technologies. Your Dashboard can tick at time intervals or once an event comes in. Be creative and come up with nice dashboard to display.

a. Quote status
b. Bid and Ask
c. Any other values important for dashboard.
d. Any chart /graph which you can think of.

A very simple CLI dashboard is given below for your reference:

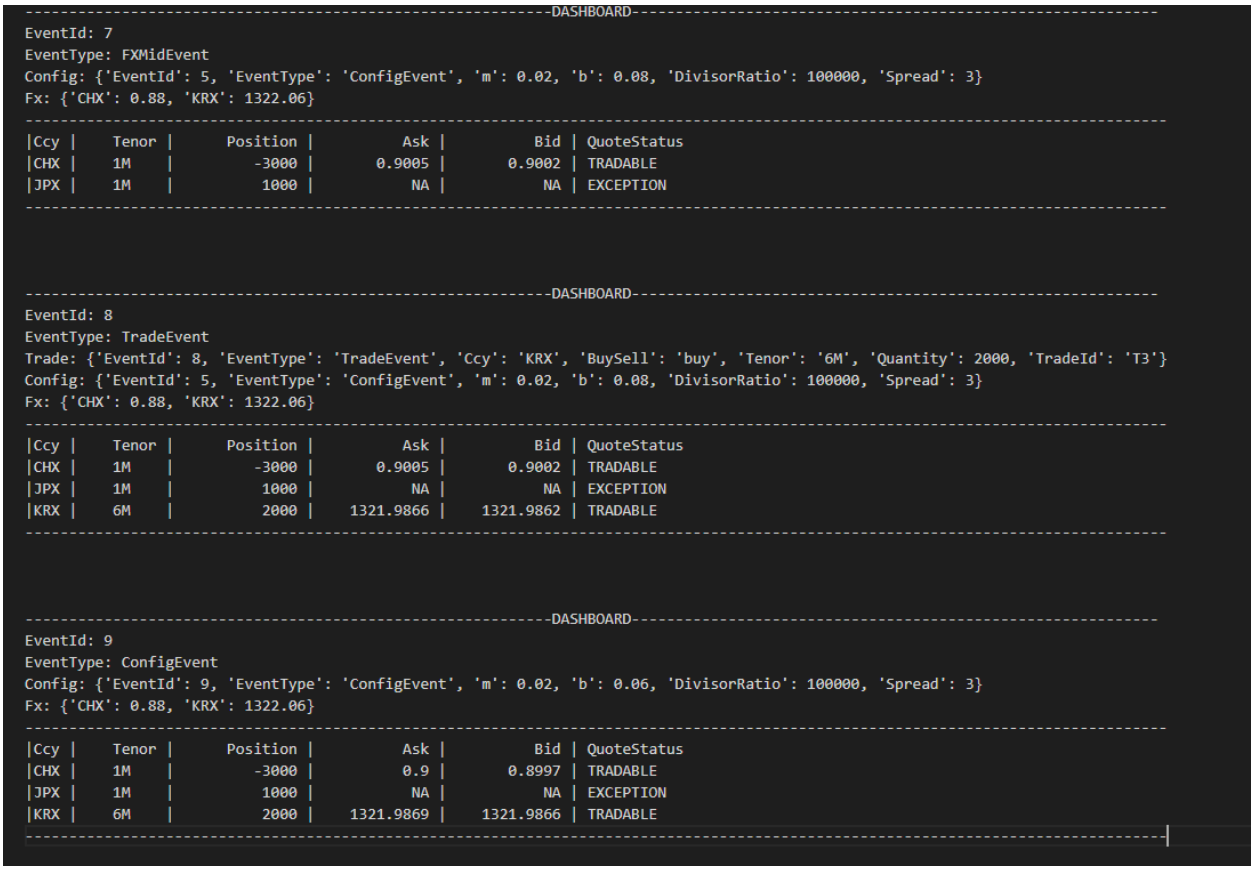

Table 9:Sample CLI Dashboards

5.2.3 Presentation

The following is a suggested format of your presentation. This is for your reference only. You will get only 5 minute to do your presentation.

1. Cover Page: include your team's name and member details
2. Overview: describe the high-level approach taken for the solution and components you have attempted.
3. Design and architecture of your solution.
4. UI interface demo/ screenshots
5. Optionally, any additional commentary on your solution
6. Takeaways and conclusions

6 Glossary of Terms

Terms	Explanation
FXForward	An FX forward is a contractual agreement between the client and the bank, or a non-bank provider, to exchange a pair of currencies at a set rate on a future date.

Bid	Price at which buyer agrees to buy a product
Ask	Price at which a seller agrees to sell the product
Spread	Difference between Bid and Ask
Positions	Total quantity group by Ccy and Tenor (for our use case)
Tenor	Tenor refers to the length of time remaining before a financial contract expires
Currency	Currency in which trade gets represented. In real world it is USD, SGD etc.
FXRate/Rate	Rate to convert one currency to another

7 Words of Advice

This project is intended to give you an exposure into how engineers work at the bank i.e., you get to work on everything which you like, where you feel challenged and where there is immense learning. Divide the work within your team as per different components and individual strength. The UI should be functional and intuitive rather than beautiful. Try to work on simple and correct solution first and then improve it further. Moreover, an important part of working at the Bank is making it easy for the users and fellow engineers, so write very well documented code along with any supportive documentation. All the very best and make sure you enjoy the project!