BiLira Algo Trader Challenge

Instructions

Your goal is to create a web service that provides quotes for digital currency trades using data from the FTX orderbook.

For Market Orders

Exchanges maintain an order book for each tradable currency pair (eg. BTC-TRY). An order book consists of a series of bids (offers to buy) and asks (offers to sell). Bids are sorted descending by price (highest price first) and asks are sorted ascending by price (lowest price first). If these two numbers ever cross, a trade is executed and those bids or asks are removed from the order book. The FTX API exposes an endpoint to retrieve the current order book for each currency pair. For this task, you should use the orderbook endpoint to fetch aggregated order information.

Your service will handle requests to buy or sell a particular amount of a currency (the base currency) with another currency (the quote currency). The service should use the orderbook to determine the best price the user would be able to get for that request by executing trades on FTX. Note that the quantity your user enters will rarely match a quantity in the order book exactly. This means your code will need to aggregate orders in the order book or use parts of orders to arrive at the exact quantity, and your final quote will be a weighted average of those prices.

For Limit Orders (Optional Question)

Your service will handle requests to buy or sell a particular amount of a currency (the base currency) with another currency (the quote currency). For limit orders, the service does not

need to retrieve the order book information as the order is placed where the user enters the price, so you don't need to calculate where the order will be executed, just return where the order or orders will be placed.

There is an additional parameter, number of iceberg order. An iceberg order is an order to buy or sell a large quantity of a financial security that, rather than being entered as a single, large order, is broken up into several smaller orders. If the user specifies it 1, there will be no iceberg order, so your system should return a JSON Array with only one order with the relevant information. If the number of iceberg order is greater than 1, your system should split the quantity into equal parts and return a JSON Array containing the specified number of orders.

Service Specification

The web service should have two different endpoints one for limit and one for market orders, that receives JSON requests and responds with JSON. If there are any errors processing the request, it responds with a JSON object including an error message.

Your service should only support spot (not futures) markets on FTX.

The service should be able to quote trades between any two currencies that FTX has an orderbook for. It should also be able to support trades where the base and quote currencies are the inverse of a FTX trading pair. For example, the service should be able to quote a buy of USD (base currency) using BTC (quote currency), even though the FTX orderbook is BTC-USD.

Route: POST /quote

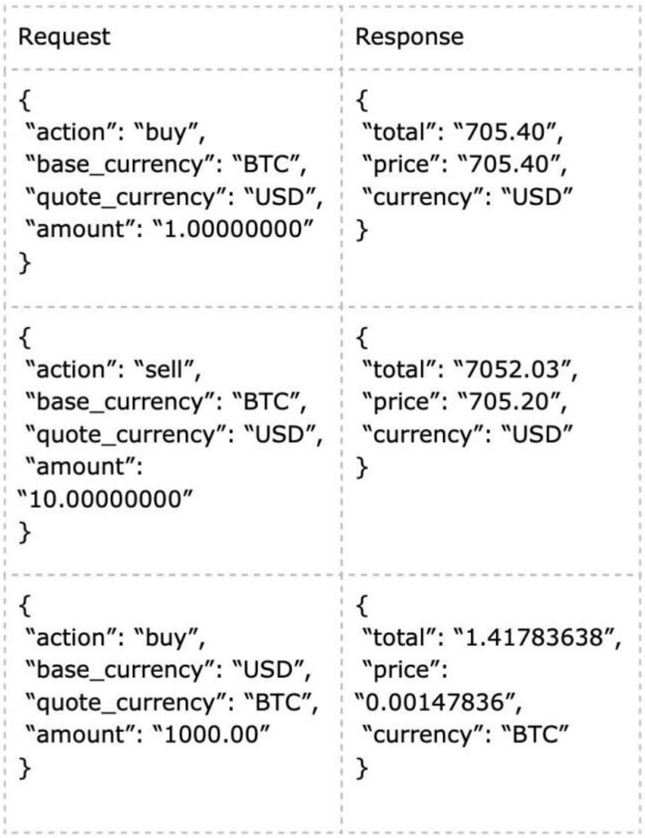
Market Order Request

- action (String): Either “buy” or “sell”
- base_currency (String): The currency to be bought or sold
- quote_currency (String): The currency to quote the price in
- amount (Float): The amount of the base currency to be traded

Market Order Response

- total (Float): Total quantity of quote currency
- price (Float): The per-unit cost of the base currency
- currency (String): The quote currency

Examples



Request	Response
{ “action”: “buy”, “base_currency”: “BTC”, “quote_currency”: “USD”, “amount”: “1.00000000” }	{ “total”: “705.40”, “price”: “705.40”, “currency”: “USD” }
{ “action”: “sell”, “base_currency”: “BTC”, “quote_currency”: “USD”, “amount”: “10.00000000” }	{ “total”: “7052.03”, “price”: “705.20”, “currency”: “USD” }
{ “action”: “buy”, “base_currency”: “USD”, “quote_currency”: “BTC”, “amount”: “1000.00” }	{ “total”: “1.41783638”, “price”: “0.00147836”, “currency”: “BTC” }

In the previous example, the trade would be executed against the BTC-USD order book despite the base currency and quote currency being reversed.

Route: POST /quote **Optional Question**

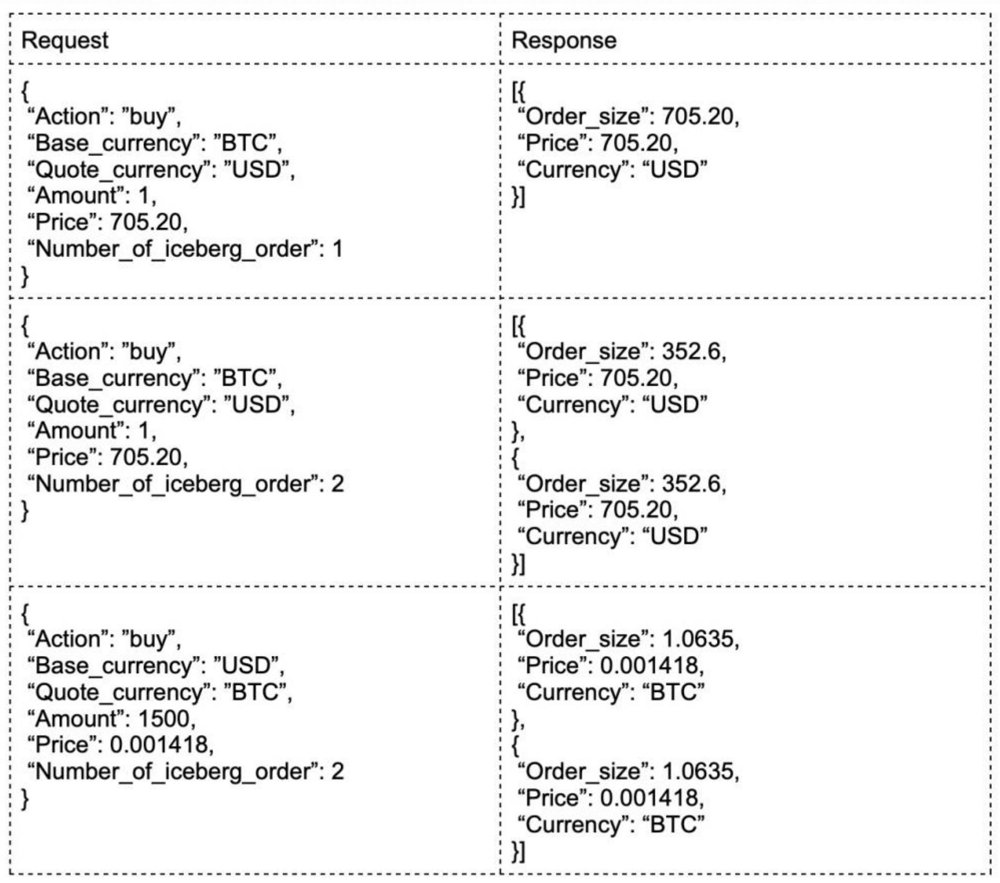
Limit Order Request

- action (String): Either “buy” or “sell”
- base_currency (String): The currency to be bought or sold
- quote_currency (String): The currency to quote the price in
- amount (Float): The amount of the base currency to be traded
- price (Float): The price of limit order
- number_of_iceberg_order (Integer): The number of iceberg orders between 1 to 5

Limit Order Response

- order_size (Float): Total quantity of quote currency
- price (Float): The order price
- currency (String): The quote currency

Examples



Request	Response
{ “Action”: ”buy”, “Base_currency”: ”BTC”, “Quote_currency”: ”USD”, “Amount”: 1, “Price”: 705.20, “Number_of_iceberg_order”: 1 }	[{ “Order_size”: 705.20, “Price”: 705.20, “Currency”: “USD” }]
{ “Action”: ”buy”, “Base_currency”: ”BTC”, “Quote_currency”: ”USD”, “Amount”: 1, “Price”: 705.20, “Number_of_iceberg_order”: 2 }	[{ “Order_size”: 352.6, “Price”: 705.20, “Currency”: “USD” }, { “Order_size”: 352.6, “Price”: 705.20, “Currency”: “USD” }]
{ “Action”: ”buy”, “Base_currency”: ”USD”, “Quote_currency”: ”BTC”, “Amount”: 1500, “Price”: 0.001418, “Number_of_iceberg_order”: 2 }	[{ “Order_size”: 1.0635, “Price”: 0.001418, “Currency”: “BTC” }, { “Order_size”: 1.0635, “Price”: 0.001418, “Currency”: “BTC” }]

Docs

- FTX API: API Documentation

Submission

You can use any language and framework of your choice, we recommend you choosing the ones you are most proficient in. The only requirement is that it can be set up and run on a Unix environment.

When you are done, email back a Dropbox or similar link to your source code and instructions to run the project in a Unix environment as well as how much time you spent on the problem. If there are any issues during setup, we may reach out to you.

Evaluation

Your submission will be evaluated on:

- Adherence to specifications
- Code quality, documentation and readability
- Idiomatic usage of the language or framework of your choice
- Data structures, design patterns, and architectural tradeoffs
- Handling of edge cases and errors