ALGOMI

ALFA

ALGOMI ALFA API

We are excited to announce the release of the Algomi ALFA API (Application Programmable Interface). Helping Fixed Income Portfolio Managers, Traders, Quants & Engineers access real-time normalised market data – in a few lines of code

FOCUS ON GENERATING BUSINESS VALUE

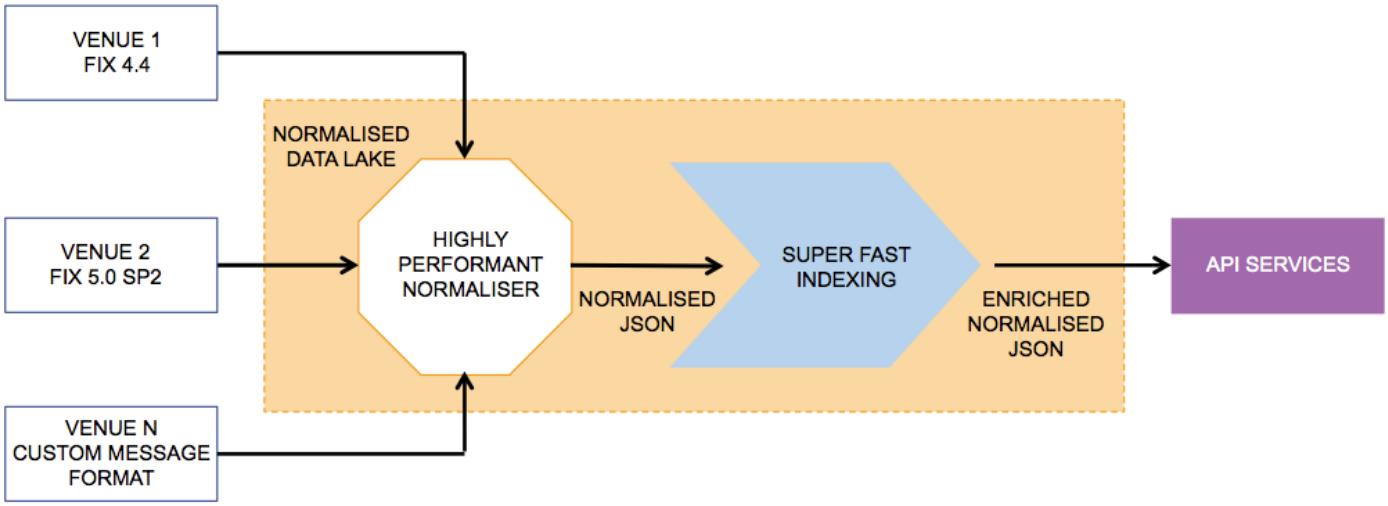
Many people in Fixed Income have expressed interested in learning Python, Java, Node JS (Java Script) etc. Why? Because they are looking to empower themselves with tools that can help them source and analyse huge amounts of market data, enabling them to make more informed investment decisions. Now you can.

Learning Python is one thing, but building a technology stack that can consume huge amounts of market data, clean it and make it easily accessible for analysis requires a vast investment of time and money, both of which are hard to find in a continuously time and cost constraint world. To pull off a project that involves building such an infrastructure, one needs budget approval, project management, engineers (GUI and server), testers, designers, devops, support and ops, the list goes on. Before writing a line of code, hundreds of thousands of dollars are spent just on planning alone. It does not have to be that way.

At Algomi, we have created a platform that takes care of this, saving you time and money, so you can focus on making that data work to generate business value.

THE TECHNICAL BIT

The Algomi ALFA platform, which is deployed on a client's cloud instance, consists of a highly performant and scalable real-time micro-service architecture which can consume large volumes of messages from the client's connected venues and within nano-seconds, convert messages in disparate formats to a normalised JSON (Java Script Object Notation) format. Once normalised, the JSON messages are indexed in real-time using Algomi's proprietary super fast and highly scalable zero-contention indexing engine which enriches the normalised JSON messages with a unified identifier (typically ISIN), allowing for easy querying.



Up until recently, these enriched JSON messages were published to an internal message bus used by the Algomi ALFA GUI (Graphical User Interface). Now, excitingly, these same messages are also published to a websocket server which our clients can instantly access through simple code regardless of the coding language.

THE REALLY EXCITING BIT

You are sitting at your desk and you're starting up your Python editor - your Algomi ALFA API credentials have just arrived.

To connect, install the Python websocket library with this command :

```
pip install websocket-client
```

Create a file “api.py” and write the following code, Algomi can supply you with the connection logic code:

api.py — real-time-api

api.py

```python
import urllib
import urllib2
import json

import websocket

try:
    import thread
except ImportError:
    import _thread as thread


def on_message(ws, message):
    print(message)


############ Algomi supplied connection logic below  ############

def fetch_access_token():

    #####logic to fetch security token - Algomi supplied

```

The “message” on line 14 contains the real-time normalised enriched JSON message, which is the result of all the powerful computation described in the 'THE TECHNICAL **BIT'** sectionabove.

A redacted version of the “message” looks something like this:

```
alfaMessage:
     {
       alfaId: 'X1225433AK71'
       timestamp: '2019-01-21T14:22:04.170Z',
       action: 'Bid/Offer',
       bidPrice: 100.00,
       bidSize: 1000000,
       offerPrice: 100.25,
       offerSize: 1000000,
       direction: 'Both',
       firm: 'XX',
       source: “SOURCE X”
       isAxe: 'N',
       type: 'Quote'
       ...
}
```

The alfaId is the enriched identifier which maps custom venue bond identifiers to a familiar id e.g. ISIN in this case. It also contains pricing information provided by your sources.

EXAMPLE USE CASE

Say you want to be alerted when a particular bond is approaching a target offer price from one or more firms – the logic is rather simple:

```
17  if message.alfaId == myFavBondIsin:
18      if message.offerPrice == myTargetOffer:
19          print('Target Offer seen for',myFavBondIsin , ' at firm: ', message.firm)
```

You can use the alfaId and offerPrice in the “message” to write this piece of logic. The print function will output the alert to your code editor. On top of this, you can import an alerting function (typically approved by your firm) that can text/email/chats this alert.

This is just scratching the surface.

Coding languages have a vast array of libraries (re-usable pre-written code) that can be used to create much more sophisticated logic, which in turn can generate more value adding business outcomes. For example, using the "message" from above, you could code:

- **Real-time Technical Analysis**

  Use an in-memory data structure to process bond prices/sizes extracted from the “message”. At an interval, calculate the *relative strength index (RSI)* in real-time. This could be used for break-out analysis, generating alerts when a price passes through a support or resistance line.

- **Supervisory Machine Learning (ML) bond price sentiment**

  Use bid/offer prices extracted from the “message”, seen on individual venues as weighted feature vectors. These can be cross-referenced with historically executed prices, quantifiable research, user behaviour and events etc. You can start with a basic *linear regression model* to create a hypothesis function, which can calculate a propensity-to-trade analytic per venue. This can be further enhanced using more powerful ML algorithms like *Support Vector Machines* or *Random Forest*.

- **Anomaly detection for compliance**

  Bid/offer prices extracted from the “message” can be compared, in real-time, against the executed price to determine if the price is above or below compliance parameters or thresholds. This can help generate up to date snapshots of the market to help qualify best execution or assist with pre-trade transaction cost analysis.

There are many more exciting use cases we are working on. However, you know best what business outcomes you want to achieve using normalised real-time data.
Now you have the API to do that.