BROADPEAK PARTNERS, INC.

Price Movements Using K3

Client: Global Hedge Fund

Price Repository: LIM

Total Price Quotes: Approximately 450 (Gas, Power, Oil, Refined)

Project Duration: 15 Days

Parallel Testing of all Price Quotes: 15 Days

Overview

Client uses LIM as a "gold source" of price information. All applications requiring price must integrate to LIM in order to diminish the possibility of "multiple versions of price truth." The client has multiple downstream systems requiring price including:

Trading System (Java system requiring prices to be presented in Soap)

Risk Repository (C# /Oracle application requiring prices to be presented in XML)

Margin System (Java requiring prices to be presented in JSON)

Approach and Implementation

K3 has an established connection to the LIM API. This consists of a request component that sends a Soap message over HTTPS to the LIM API. In order to receive a response K3 must send the request with in both the correct format (SOAP via HTTP) and the correct mapped information (For LIM 3 data elements are required: 1) Execution Units, 2) Column and 3)Symbol.)

For each required price t K3 pulls the mapped data (the 3 data elements) and calls the Web Svcs component in F1. This component converts the information to Soap over HTTPS. This then coordinates with the routing component to send the request to LIM. (Note: The K3 LIM connection is designed to execute in batch mode allowing multiple quotes to be retrieved at once.)

When this formatted message is presented to LIM, it returns a response in Soap over HTTPS.

Once the Soap/HTTPs message is sent by LIM and received by K3 it is immediately persisted.

Transformation To Target

When the Soap/HTTPS message is received by K3 it is converted to a normalized state. This is performed by calling the JAXB component in F1 to convert the message to a simple Java object. The reason for this first conversion is to normalize the message so it can be easily converted to any other format to feed to one or multiple downstream systems.

To Downstream Systems

K3 supports connections to an unlimited number of downstream systems. Here we have 3 connections for price movement. These are: (K3>Trading System), (K3> Risk Repository), and (K3>Margin)

As you will recall the system is has saved the original SOAP/HTTPS response into a normalized Java component. K3 executes a multiplexing function so that the following functions happen simultaneously. Multiplexing is a standard component in the F1 framework. By calling this component the original normalized message is split into 3 messages and *simultaneously* goes through the following:

To the Trading System.

1. Pull the mapped data from K3 to generate acceptable values for the trading system. For example, LIM knows a quote as (Execution Units, Column and Symbol) but the trading system knows a quote as (Quote Name, Quote Number). Here K3 looks up the correct domain mapping and converts the original LIM values to the trading system values.
2. Second, K3 takes one arm of the multiplexed Java message and calls the Web Services component and re-converts it to SOAP/JMS.
3. Third, the message is routed via the K3 routing component to the trading system's API

To the Risk Repository

1. K3 pulls the mapped values to translate them to target system values.
2. K3 takes one arm of the multiplexed, normalized message in Java and calls the JMS/XML components to convert it to the correct format
3. Finally K3 leverages the routing component to route to the risk system

To the Margin System

1. K3 pulls the mapped values to translate them to target system values.
2. K3 takes the last arm of the multiplexed, normalized message in Java and calls the JSON component to convert the values to the correct format.
3. Finally K3 leverages the routing component to route it to the Margin System.

For Business Users

One of the most important things for this client was the ability to:

- See what data is flowing through the interface and easily validate that it has hit its target
- Provide the ability to continually change mapped values between systems
- Enrich data as appropriate

Feed Manager

In terms of viewing data, this is about operational control. The primary reason for using K3 was that if prices were not fed correctly, it was not known until the end of day failed. This requires a great deal of end of day manual effort. However, here all prices can be seen as they flow through K3 to their target system. Feed Manager is also looking for a response from the downstream system. K3 automatically tells the user whether each price has successfully found its target in downstream systems. To further this purpose, we have enabled K3 with email alerts such that if any load error fails the appropriate parties can be informed immediately.

Map Manager

The second requirement is the ability for business users to be able to easily and dynamically map values. For price integrations with LIM and Zema , the importance of this cannot be understated. Systems such as LIM and ZEMA DO NOT allow the user to use its own product codes for prices. They are held in these systems under LIM and ZEMA unique formats. Trading systems use an entirely different set of names and codes to represent price data. Thus, every price quote, whether a forward price, settle price, calculated or composite curve, requires mapping. K3 Map manager is specifically designed to support mappings in a single and cohesive framework that not only manages multiple interfaces, but is also user friendly.

Rules Manager

Finally, on occasion, certain systems require special rules. For example in the client install, some prices such as Eurodollar Swaps come from LIM with 6 decimal places. For business reasons the client wanted these prices rounded to 2 decimal places. Thus we created a rule: If product =Eurodollar then Round to 2 decimal places.

Conclusion

In summary the client was able to create an automated feed from LIM to all of its downstream systems in less than a month. All price changes in LIM automatically flow to their destination. Every

passed data element is able to be viewed and validated by client users. Likewise as the business expands, adding new products is facilitated by a user friendly interface to add and change values.

We estimate that the client saved over 50 person hours per month in terms of eliminated manual effort required from manual upload of prices to downstream systems, error chasing relative to failed loads at the end of the day and manual price revisions, and finally elimination of code modification necessary to making additions /modifications to testing and mapping.

Do you want to move prices efficiently throughout your infrastructure? Call us.

Phone: +1.646.370.6804

Email: info@broadpeakpartners.com