

# Challenges faced by using FIX for FX Streaming Prices

By Daniel J. McCarthy, Development Manager, Global Link, State Street Global Markets



With the formation of the FX Working Group for FIX, there's been a lot of discussion generated recently about using FIX for streaming prices (and FX in general). As with any standardization effort, there are inherent challenges to overcome, including gaining consensus and 'normalizing' what the early adopters have, as well as describing and enabling any special workflows, while also allowing bank specific customization for particular stream attributes.

## Opportunity

While there are some FX trading systems that currently have dealable streaming functionality, the protocol choice for these systems is generally a proprietary one. A huge opportunity exists in the integration of aggregated market data, directly into buy side order management systems. Buy side users will benefit by giving their trading models 'plug-in' access to executable FX streams. The ideal protocol choice for the market data component would be open and flexible, and something that existing OMS systems already understand - FIX. Most order management systems already have execution capability in one or more asset classes based on FIX, so extending the reach of the OMS to streaming market data via FIX is a logical next step. These market data interfaces have perhaps already been defined and implemented for other asset classes by using vendor specific protocols, but there is an additional opportunity to create a global standard using FIX for transmitting and dealing on market FX rates. You might ask why wouldn't one continue with the market data integration that OMS already utilizes, but by using one protocol such as FIX, one can ease validation by creating a linkage (i.e. Quoteld) between the streaming market data feed and the customer acceptance of the rate. This is perhaps impossible using vendor specific protocols.

## Technical Challenges

In each asset class, streaming market data presents a different set of challenges. In Fixed Income securities, there is virtually no central marketplace, so pricing information is difficult to obtain. The concern is where to find a particular security that meets buy side requirements, and then negotiating a fair price. The universe of fixed income securities is vast, which generally creates an identification challenge (CUSIP, SEDOL, ISIN, etc), but market data updates would presumably be relatively infrequent when compared to other asset classes. By contrast, the equity arena is centralized around exchanges (and ECNs), and while the universe of stocks is large - many are very heavily traded. The combination of a large number of stocks, frequent price updates, and a large body of subscribers creates a bandwidth issue for efficient dissemination of equity market data. Market data vendors have addressed these issues, but their protocols are proprietary, and thus any integration with buy side execution can become an expensive endeavor.

On the surface, implementing a streaming market data feed for foreign exchange looks straightforward. Instead of exchanges as in the equity model, banks provide quotes, but basically it is still a central pricing source, with each bank considered a different exchange (or a market maker if you

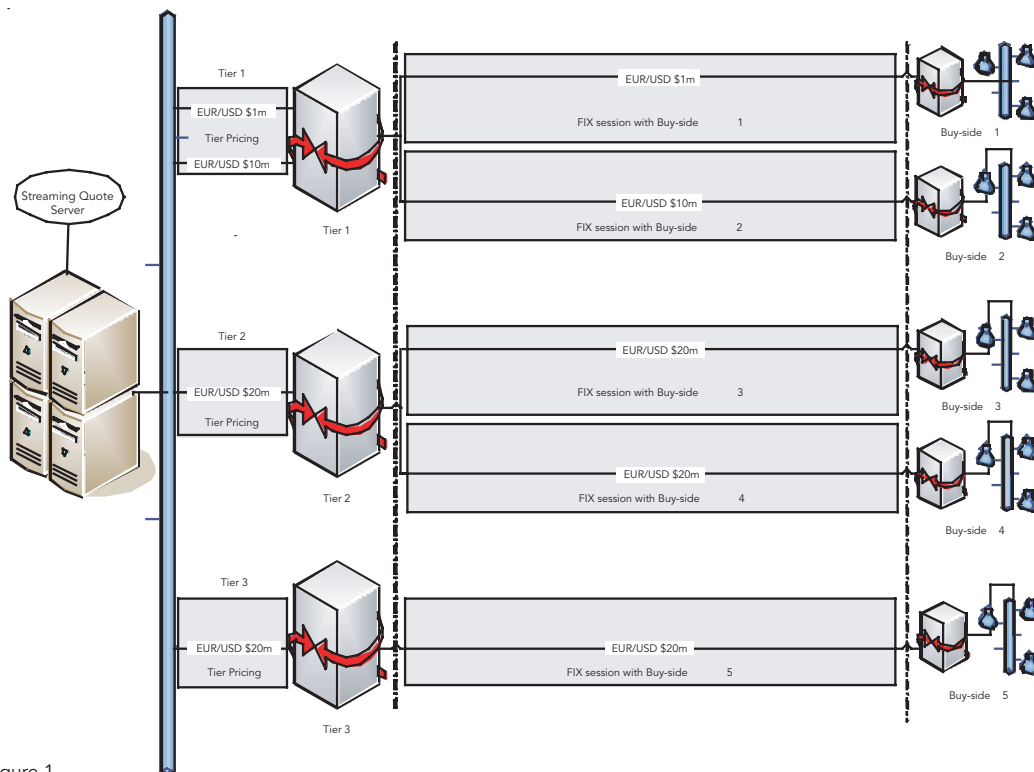


Figure 1

prefer a NASDAQ analogy). The universe of currencies is relatively small, especially when compared to the number of equities and fixed income securities. FX pricing can be extremely volatile, especially for the most actively traded currency pairs, but since there are fewer of them, the total bandwidth requirements are not as daunting as equities. However, there are a few wrinkles that keep this from being simple. First, banks want to create separate 'volume bands' for each currency pair so their quotes are contingent on the requested amount. From the bank perspective this is logical, since larger deal sizes generally translate into increased risk. Essentially, this partitions the quote stream for each currency pair into channels which need to be treated by the downstream consumer systems as distinct. In addition, some banks will want to further stratify this by grouping their clients into tiers. The combination of volume bands and tiering has a multiplicative effect on the number of messages per second that the banks automated pricing server must handle. Figure 1 (Tier 1) shows an example of two different 'tier 1' buy sides requesting a quote for the same currency pair (EUR/USD), but for different amounts.

These additional requirements serve to create a potential bandwidth issue. For example, assume a bank

will provide streaming prices in 25 currency pairs across 6 volume bands, and that each currency pair updates once per second. Assuming at least one active subscriber in each volume band, and the bank's pricing server will need to be capable of providing  $(25 \text{ CCY pairs} * 6 \text{ bands} =) 150$  messages per second. If that bank also wants to classify its customers into 4 tiers as well, then the workload on the pricing server increases to  $(25 * 6 * 4 =) 250$  messages per second. Of course, that is worst case scenario, with subscribers for each currency pair and all combinations of volume band/tier have at least one active subscriber.

From the buy side perspective, even though users will likely only be allowed to subscribe to one volume band per currency pair at a time, they will undoubtedly wish to subscribe to rates from several banks at once across multiple currency pairs. The net effect from the end user perspective is roughly the same  $(25 \text{ CCY pairs} * 6 \text{ banks} = 150 \text{ messages per second})$ , even though the load is spread across multiple FIX sessions. The FIX traffic on any individual FIX session will be driven by the number of currency pairs that the buy side users are subscribing to at any given time (up to 25 CCY pairs in our example).

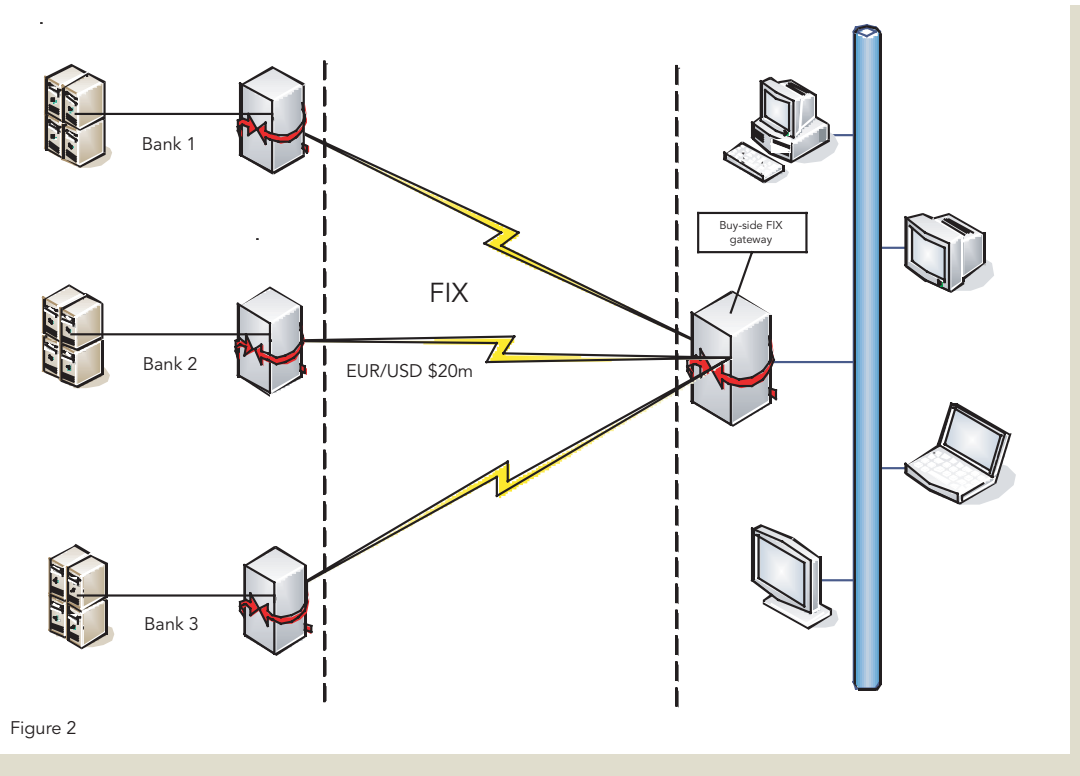


Figure 2

If the assumptions are correct, then the main area of concern is how to handle the volume of traffic to and from the FIX engine.

### Possible solutions

Most FIX engines today can comfortably handle volumes in the hundreds of messages per second, and the potential bottlenecks exist not at the FIX session, but the internal interface point. One possible solution to this is to use a multicast protocol on both the bank and customer sides of the FIX session. Unfortunately, FIX sessions are 'point-to-point', so it isn't possible to multicast directly from bank to customer.

On the bank side, a multicast protocol would manage the transport of market data from the central point of creation (pricing server) to individual FIX sessions. The benefit of the multicast solution is to organize published market data information into a series of one-to-many relationships. Each active combination of currency pair and volume band (and tier if applicable) would form a distinct channel, and receive the same set of ordered messages (price ticks). In other words, a market data channel could be shared by multiple FIX sessions (See Figure 1, Tier 2). In this case, the quotes provided to buy side 3 and buy side 4 would be the same. Buy side 5 (Figure 1, Tier 3) is requesting the same currency

pair, and the same amount, but they might get different quotes, since they are a tier 3 client.

On the buy side, multicast can also be employed to manage the transport of market data from multiple FIX sessions, aggregating 'streams' from all banks into a single client GUI (see Figure 2).

Another possible solution would be to drop the requirement for FIX sessions, and multicast directly from bank to customer using FIXML messages. In order for this to be feasible, the banks and customers would all need access to a shared network. This would provide the most efficient method of delivering the market data, but the design of such system would need to include the functionality currently provided by the FIX session layer. Interestingly enough, the FIX Market Data Optimization Working Group currently has an agenda item to address the multicast requirement for all asset classes. In either the FIX or FIXML case, multicast is certain to be part of the overall solution. **FIX**

### Any thoughts on this or other articles?

Please send any comments, referring to this article as Vol 1 Issue 5 GL3, direct to Edward at [edward@fixglobal.com](mailto:edward@fixglobal.com)