Introduction

The TIBCO FX Dealing Accelerator is a reusable set of software components that provide TIBCO FX customers a fast start to deploying FX Market Data and FX Dealing solutions. The FX Dealing Accelerator is available to TIBCO customers as a package of TIBCO StreamBase EventFlow™ source code with TIBCO® Live Datamart table definitions and a sample JavaScript user interface, to be used as a starting point for your customization. The primary components of the FX Dealing Accelerator are:

- A working subset of the TIBCO Trading Components Framework that provides FX venue connectivity, venue handlers and execution handlers. Using TIBCO StreamBase Studio™, connectivity to FX venues is achieved by drag and drop functionality. The associated venue handler is automatically configured and is ready to consume fast market data.
- FX Pricing Aggregation: As market data is ingested from the configured venues, it is passed through an aggregation interface that generates a price. This price is used for tiering and customized spreading. Initial spreads are loaded at startup but can be changed at runtime, enabling dynamic price calculation.
- Pricing subscriptions are registered by the Accelerator and are serviced at specified intervals. This guarantees full control of downstream message rates.
- Price Distribution: Prices are distributed via a TIBCO® Live Datamart interface. This provides ad-hoc query support for the FX Dealing user interface. The prices are also available to the Live Datamart JavaScript API, and a sample HTML5 trading user interface is included. (Other distribution options include: TIBCO EMS, TIBCO FTL, TIBCO RV, Thomson Reuters RMDS, Solace Systems, Tervela, Wombat, or 29West.)
- Execution Handling: Trade reporting and Real Time Profit and Loss are implemented, and this data is available in the provided Live Datamart data tables and in a sample HTML5 user interface. Positions can be monitored and also sent downstream for clearing and further processing. (Trades are executed against a simulator. When extending the accelerator to a live environment, you must remove the simulator and route the trades to a live execution handler.)
- Sample JavaScript Pricing and Trading User Interface: The FX Dealing user interface is a sample user interface for FX Dealing, built in JavaScript and HTML5. The sample provides a visual representation of aggregated FX market data, FX price spreads, top of book, and a point-and-click interface to set FX prices for an FX dealer. The user interface is intended to be extended by TIBCO customers and is provided as a fast start.

Running The FX Dealing Accelerator

1. Open the Studio Preferences dialog. In the menu:StreamBase Studio[Test/Debug] panel, make sure the Try another random port option is selected.

   (On Windows, open the Preferences dialog with menu:Window[Preferences] . On the Mac, use menu:StreamBase Studio[Preferences] .)

2. Run the Accelerator as a Live Datamart project. This is done in one of two ways:

[loweralpha].. **From the LiveView Project Viewer**: The Project Viewer opens by default when you load the Accelerator into Studio. Select the tab for the Project Viewer, which bears the name of the project as seen in the Package Explorer view. Click the green Run button in the upper right corner of the Project Viewer.

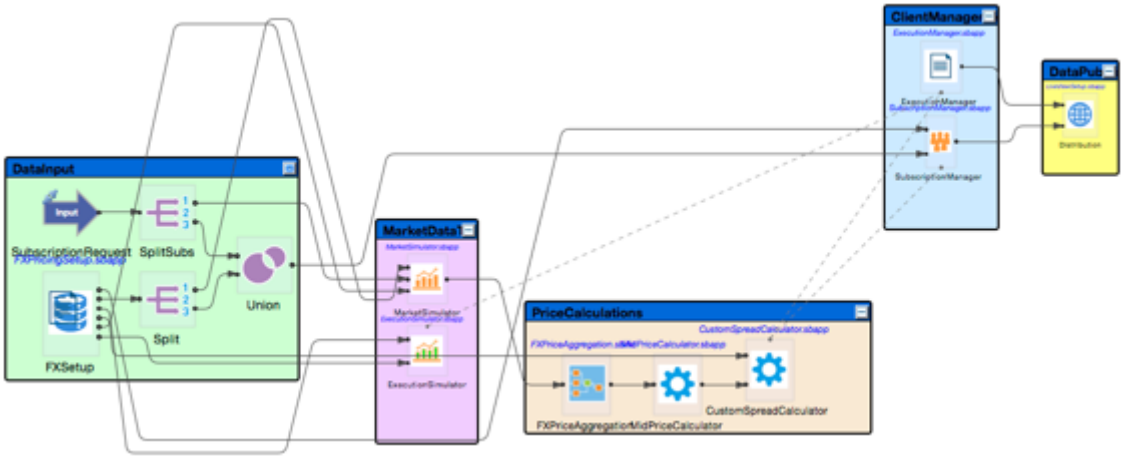
(If the Project Viewer is not open, right-click the project's name in the Package Explorer view and select Open LiveView Project Viewer .) .. In the Package Explorer view, select the *sample_Frameworks_TradingComponents_fxpricing* project, right-click, and from the context menu, select menu:Run As[StreamBase LiveView Project] .

3. The Console view shows several messages as the LiveView Server compiles the project and starts. Wait until the following console message appears before proceeding to the next step: `All tables have been loaded. LiveView is ready to accept client connections`.
4. Open a web browser and navigate to *http://localhost:10080/angular-sample/FX_Dealing_Accelerator.html* .
5. In the Package Explorer view, open the *sample_Frameworks_TradingComponents_fxpricing* project, then double-click to open the *FXPricing.sbapp* . Make sure the application is the currently active tab in the EventFlow Editor.
6. From Studio's top-level toolbar, click the Run button. This opens the SB Test/Debug perspective and starts the application. You can send test tuples to this application in the Manual Input view. Tuples sent connect to the Live Datamart server in query form.
7. In the Application Output view, observe tuples emitted on the `LVStatus` stream, which give details about the connection to the Live Datamart server.
8. When done with the StreamBase application, press kbd:[F9] or click the Stop StreamBase Application button to stop the *FXPricing.sbapp* application.
9. When done with the Live Datamart application, press (on the Mac) or click the red and blue Stop LiveView Server button to stop the Live Datamart application.

FX Pricing Using VWAP Trading Components Sample

* About The Sample*

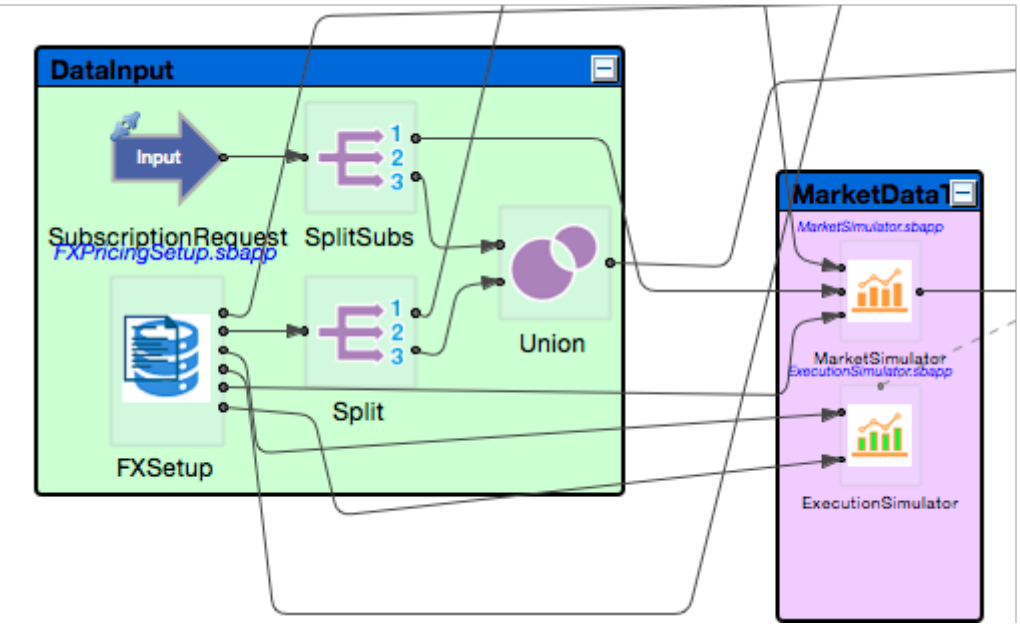
This sample demonstrates how to generate custom Bid-Ask price ladders from Blended Depth of Book market data using the Volume Weighted Average Price. The sample also features a Subscription Manager that can be used to send data downstream to clients at specified time intervals and an Execution Manager that performs real time Profit/Loss calculations. In this sample application we have two FX Market Data providers configured: FXSpotStream and Deutsche Bank AutobahnFX Rapid.



Reading the sample from left to right, we have an initialization phase containing the module reference **FXSetup**. In this module the application loads:

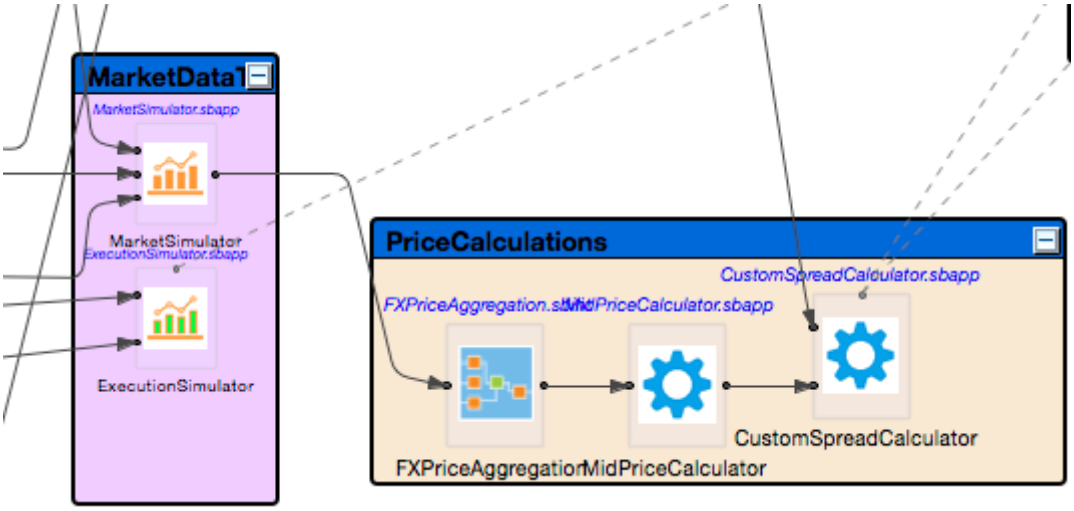
- **inital_spread.csv**: This file contains the initial spread settings for each customer Tier and Volume Band. An example row is: **EUR,USD,A,1M,4.** This tells the spread calculations to apply 4 pips (pip = 1/10000th unit) to the current market mid price.
- **initial_subscriptions.csv/preferences**: This file contains a list of subscriptions for a set of currency pairs, and a determined Tier. This means that different subscriptions can have, if desired, different spreads for the same currency pair. The preferences file also specifies a desired publishing rate.
- **initial_executions.csv**: This file contains a set of trade execution messages that can be used to pre-populate the application at start up. If this is not desired then the contents of this file should be deleted at startup.

Continuing right to left the subscriptions are sent to the **MarketSimulator** and the **SubscriptionManager**. The market simulator will now generate price data for the currency pairs requested.

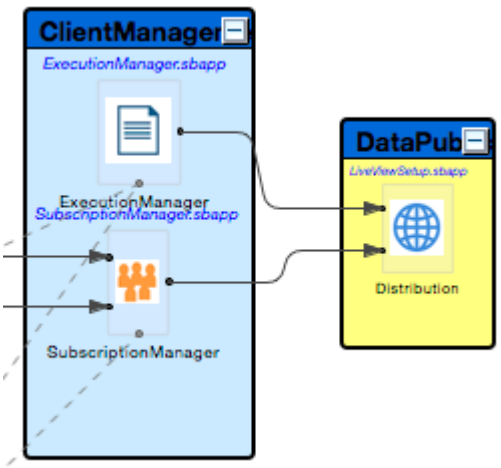


Moving onto the **PriceCalculation** we meet an example of an **ExtensionPoint, FXPriceAggregation.sbint.** This is an interface with the implementation in this case of a VWAP

calculation to obtain an weighted average of the current market data update message. When using this sample to build more complex trading systems, this implementation can be replaced or augmented to achieve the desired method of price generation. This price forms the bases of the Tiered prices that are derived in **CustomSpreadCalulator**. In this module prices are generated in compliance with the spread that were loaded at start time. It's important to note here that these spreads may be changed at runtime by enqueuing an event to *FXPricing.WorkingProcessor.FXSetup.ManualSpreadUpdate. * When this event is sent, the spreads for the currency pair are re-calculated, and will be published subsequently.



In Client Management there are basic implementations of real time Profit and Loss, and Subscription Management. **ExecutionManager.sbapp** provides an example of how to calculated profitability on trades executed real-time, and updated as each market data tick is processed. **SubscriptionManager.sbapp** a simple example of how subscriptions are cached and serviced on market data events. A suggested evolution of this module would potentially use heart beating with the Client Subscription to manage the subscription life cycle.



The final stage of our sample application is the output stage. In this example we use the TIBCO Live Datamart to distribute our prices to the sample JavaScript trading user interface. Once the data is published to the Live Datamart tables is is available for ad hoc querying and charting through both the JavaScript API and the LiveView Desktop client. It is intended that this user interface can be used as a starting point to build a customized trading application.

Live Datamart Table Schemas

The Accelerator uses Live Datamart data tables to hold the current prices for distribution and the current set of executed trades. There is another lvconf file in the project, *FXPricing.lvconf* , which is used as a launcher for the combined StreamBase and Live Datamart project.

The Live Datamart data tables are *MarketDataSample.lvconf* and *ExecutionDataSample.lvconf* , which are used with an aggregation table *ExecutionDataAgg.lvconf* . The schemas for the data tables and aggregation table are as follows:

Table 1. MarketDataSample.lvconf

Field Name	Data Type	Description
Time	timestamp	Timestamp when the price was generated in the Aggregator.
ClientID*	string	A unique identifier representing each client subscribing for prices.
Currency1*	string	The base currency of the pair.
Currency2*	string	The second currency of the pair.
Tier*	string	The tier the customer is in. This will determine the spread values they receive.
Volume*	string	The volume band this price represents.
Bid	double	The Bid price for this volume band.
Ask	double	The Ask price for this volume band.

_ Fields marked with an asterisk denote primary keys._

Table 2. ExecutionDataSample.lvconf

Field Name	Data Type	Description
TradeID*	string	A unique identifier representing the trade.
ClientID	string	A unique identifier representing the client whom we have traded with.
Currency1*	string	The base currency of the pair.
Currency2*	string	The second currency of the pair.

Field Name	Data Type	Description
Tier*	string	The tier the customer is in. This will determine the spread values they receive.
Volume*	string	The volume band this price represents.
Bid	double	The price at which a client can sell.
Ask	double	The price at which a client can buy.
ExercisedPrice	double	The price at which the trade was executed.
PL	double	The profit or loss achieved by the client on this trade. This is re-calculated as the market price changes.
Side	string	Either "Long" for a buy, or "Short" for a sell.
ExecutionTime	timestamp	Timestamp when the trade was accepted.

_ Fields marked with an asterisk denote primary keys._

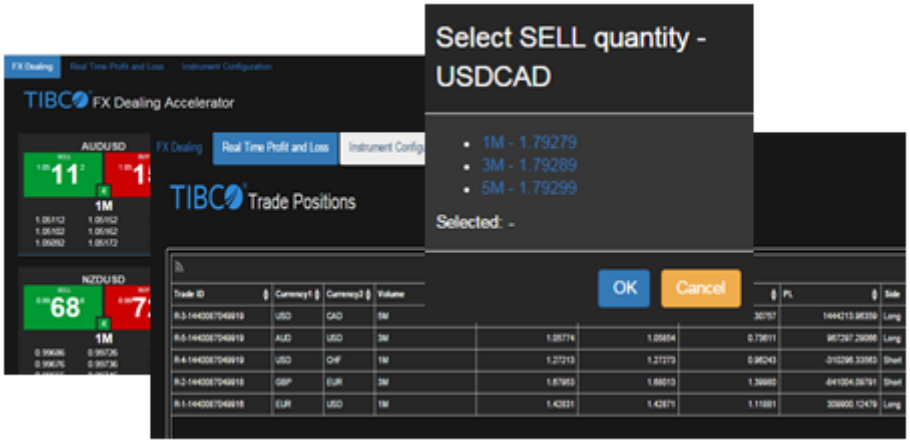
Table 3. ExecutionDataAgg.lvconf

Field Name	Data Type	Description
CurrencyPair	String	Currency1 + Currency2, such as GBPUSD.
AggQuantity	Double	Aggregate of traded quantity.
AggPosition	Double	Aggregate of traded position, such as P and L.

FX Dealing Sample User Interface

TIBCO Live Datamart provides a JavaScript API to view and interrogate the pricing and trade data. A sample Pricing/Trading user interface is provided as a fast start. The Live View Desktop is also availa2ble to create ad-hoc continuous queries and charts. Alert rule creation is achieved using the Live View Web user interface (http://localhost:10080/lv-web/index.html#/manager) or LiveView Desktop. In the sample provided there are three main screens: FX Dealing, Real Time Profit and Loss and a Configuration tab. This web user interface is built using the Component Exchange LiveView Angular Bridge and DataMart. On the main trading screen the green and red tiles are the current bid and ask prices. Clicking on these tiles pops up a modal dialog where the user selects the quantity they wish to buy or sell. These trades are then stored in the Live Datamart table

ExecutionDataSample.



Market Simulator

The sample uses the Market Simulator module to provide realistic Blended Depth of Book market data for any currency pair in the *initial_subscriptions.csv* file located in the *init_files* folder. New subscriptions can be added during runtime via the Subscription Request input stream.

Execution Simulator

The sample uses the Execution Simulator module to provide realistic Market Order Execution simulation. On start up, initial executions are loaded from the *initial_executions.csv* file located in the *init_files* folder. New executions can be added during runtime via the Execution Request input stream. The Executions table generated inside of the Execution Simulator module can be accessed outside of the handler for querying and sorting.

FX Price Aggregation

The FX Price Aggregation interface takes Blended Depth of Book market data and outputs an aggregated Bid and Ask price.

Volume Weighted Average Price

Volume Weighted Average Price (VWAP) is a method of computing a weighted average of an aggregated price book. The Bid VWAP is calculated as the sum of the products of the Bid prices with their respective quantities divided by the total quantity of Bids. The VWAP for Asks is calculated analogously. The VWAP Calculator module is an implementation of the FX Price Aggregation interface and computes aggregated Bid and Ask prices as the VWAP Bid and VWAP Ask. Note that in practice the VWAPCalculator can be swapped for any module implementing the FXPriceAggregation interface.

Mid Price Calculator

The Mid Price Calculator module takes an aggregated Bid price and an aggregated Ask price and computes the Mid Price as the average of the two. In this sample, the Mid Price is calculated as the average of the Bid VWAP and the Ask VWAP.

Custom Spread Calculator

A spread ladder is created by adding small displacements (pips) to the Mid Price based on Band and Volume. For a given Mid Price and a fixed Band and Volume, the custom Bid price is calculated as the Mid Price minus 0.005% of the pip value, and the custom Ask price as the Mid Price plus 0.005% of the pip value. The Custom Spreads table generated inside of the Custom Spread Calculator module can be accessed outside of the handler for querying and sorting. In the sample, the spread ladder is generated using the spread preferences specified in the *initial_spreads.csv* file located in the *init_files* folder. Note that in practice the Spread Preferences can be assigned via an input stream and need not be fixed.

Subscription Manager

The Subscription Manager module accepts client subscription requests for a specific currency pair at a specified Band and Volume. The Subscription Manager module also allows for personalized Update Rates so that each client can receive the data they subscribed for at their desired rate. In the sample, the Client Subscription Preferences are specified in the *initial_subscription_preferences.csv* file and the initial Client Subscription Requests are specified in the *initial_subscriptions.csv* file. Both files are located in the *init_files* folder. New Client Subscription Requests to the Market Simulator are automatically sent to the Client Subscription Manager module as well. Note that in practice the Client Subscription Preferences can be assigned via an input stream and need not be fixed.

Execution Manager

The Execution Manager module tracks current market prices for currency pairs at various Bands and Volumes against the price of an executed order. The Execution Manager reads executed orders from the Executions table in the ExecutionSimulator module and calculates real time Profit/Loss(PL) using prices generated in the Custom Spreads table in the Custom Spread Calculator module.

FX Setup

In the accelerator, the FX Setup module initializes module parameters, subscription requests, and executions from CSV files located in the *init_files* folder.