

# Deltix QuantOffice: Low Latency Productivity

*If we've heard it once from readers, we've heard it a hundred times. "How do I develop and deploy my bigger/better/smarter/sophisticated trading models quickly and efficiently?" Andy Webb, Automated Trader's Founder, takes a look at one possibility - Deltix QuantOffice.*

Automated Trader readers are a demanding lot, which is great because it always means they are hitting on us for something to do with trading. One increasingly vocal theme in their various communications over the past year has been productivity - they want more of it. That means that any trading tool that can take them from an initial idea to a deployed profitable trading model as fast as possible is highly attractive. (Take a look at this issue's Buyside Beat on page 18 for the reasons for why this is now such a priority.)

That means that apart from straightforward speed of operation and tools that expedite development, they want a holistic process; the code/language they test and develop with must be the same that they deploy - so code rewrites for a separate trading platform are a definite no-no.

## Deltix

Deltix is one of several candidates that can potentially fill this role. The company's Deltix Product Suite encompasses three major segments:

- **TimeBase** - a data warehouse that can handle multiple real-time and historic data sources that can be time series, fundamental data or news
- **QuantOffice** - a visual development, debugging and back-testing environment for integrated alpha/EMS strategies using C# and .NET

- **QuantServer** - consists of two modules (UHF Trader and Trading Console) used for live deployment of QuantOffice trading strategies. UHF Trader converts trading signals into FIX orders (or certain proprietary formats) and executes them (potentially via multiple brokers). The Trading Console is used to monitor trading activity and (if necessary) manually intervene.

Unfortunately, the Editor's continued and incomprehensible reluctance to let us use his editorial budget for live trading (plus his obsession with hoarding his precious pages) means that this review of the Deltix Product Suite will primarily focus on QuantOffice. However, we have scheduled some full and frank discussions with the Editor for after this issue of Automated Trader goes to press. We are therefore confident that after he has admired the logos on the Wrecking Crew's baseball bats we will be running a more comprehensive review of the live trading components in the Deltix Product Suite in a future issue...

## Plumbing

One of the tasks that can quickly bog down the development and deployment of trading models is plumbing. The business of connecting data feeds

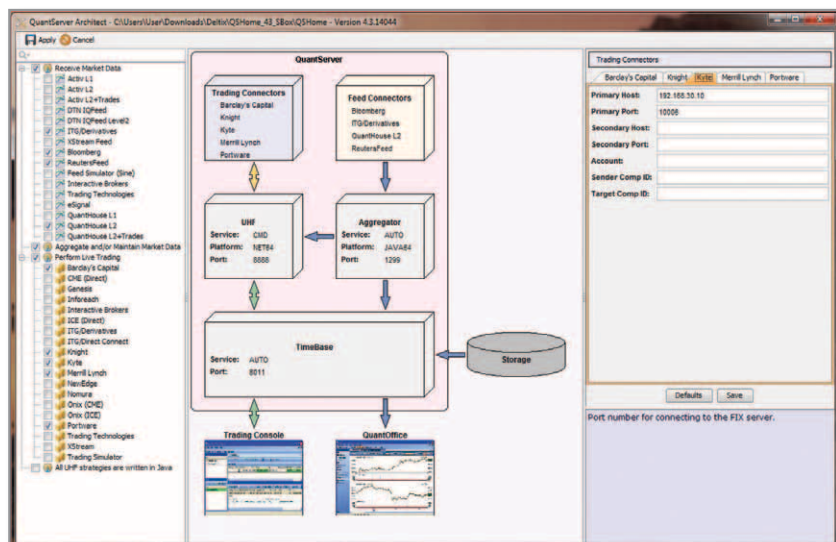


Figure 1a

Deltix QuantOffice: Low Latency Productivity

and stores to a data repository, connecting the repository to the development environment and connecting the finished models to the trading interface(s) can quickly consume time, resources and enthusiasm.

Deltix addresses this issue head on with something called QuantServer Architect. At its very simplest (and as tested in this review) it can be used to connect existing historical data to TimeBase and QuantOffice. However the fun really starts when you need to connect up live feeds and trading interfaces. By simply checking boxes you can quickly snap together all the desired connections in even complex configurations without having to write a line of code (see Figure 1a).

Clicking on a category box (e.g. Trading Connectors) in the QuantServer Architect graphic pane, opens a set of tabs in its right hand pane that allows you to adjust the relevant settings for each item in the category. For example, in Figure 1a clicking on the Trading Connectors box has opened configuration tabs for each of the trading interfaces (in this case with Kyte highlighted).

Once you have connected all the necessary services as desired, clicking the Apply button actually builds the configuration and saves it to a config file. Any problems, such as incorrect port numbers or non-running services, are automatically flagged at this stage. Figure 1b shows our finished review configuration, which highlights how the application has drilled into our storage location and identified the

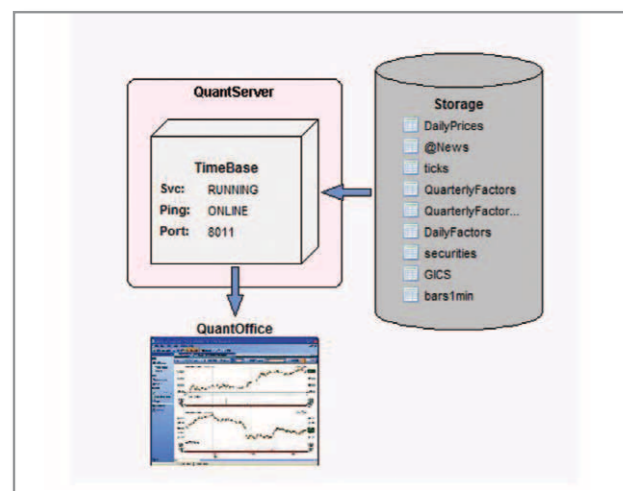


Figure 1b

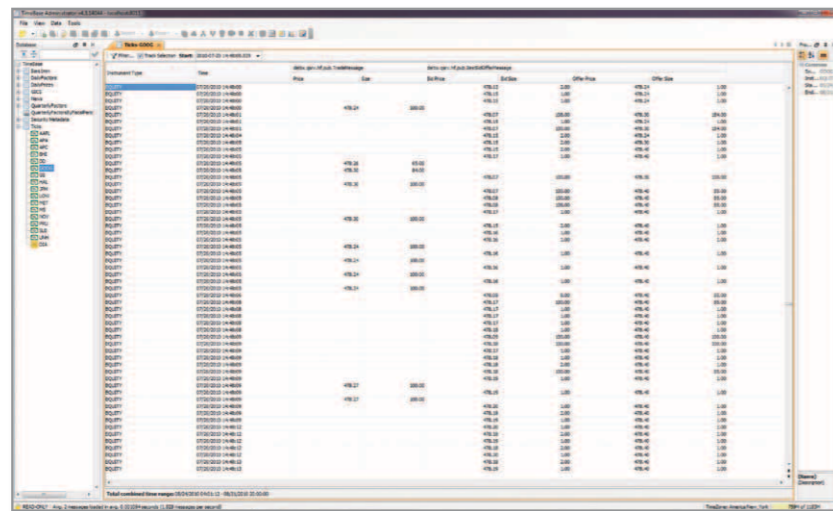


Figure 2a

various data types within it. (To be strictly accurate, in the latter stages of the review we also tacked UHF Trader onto the configuration, so we could access simulated real-time data playback for back testing).

We were only able to test QuantServer Architect in the context of the rather constrained nature of this review (we obviously plan to test the broader possibilities later) but it worked flawlessly. If the other components connect as easily, it will represent an enormous time saver when it comes to plumbing. The list of supported data feeds and trading interfaces doesn't cover absolutely every eventuality, but it does nail down a very substantial number of possible combinations.

**Data visualisation and handling**

While QuantServer Architect hugely simplifies the system integration associated with model development and employment, Deltix also provides plenty of tools to assist with delving into the nuts and bolts if desired. One example is the TimeBase Administrator (see Figure 2a), which allows for extremely granular data management. The concept of streams allows you to define precisely which data you wish to use for which task. For example, you may prefer to test a particular strategy using 20 minute bars drawn from Bloomberg data, while using Reuters data for testing another strategy using 5 tick bars or 500 share volume bars.

The Aggregator component (see Figure 1a) is crucial in this respect in that it can stream multiple real time data sources to UHF Trader for live trading as well as to TimeBase for historical storage. This opens up some interesting strategy validation possibilities if you have access to (and choose to store) multiple feeds for the same instrument. For example, are back tested results for a strategy that uses tick data identical if tested with

two different tick sources? However, the polymorphic manner in which TimeBase handles data also opens up a huge range of further possibilities when it comes to strategy development. It makes the accurately synchronised combination of data from multiple sources and of multiple types trivial, so strategies that combine news events, fundamental data, depth of book and actual trades can be easily constructed. The Administrator also allows for editing of both an overall database schema as well as individual stream schemas. That in itself is valuable, but the really neat twist is that customised stream's schemas can be created and used to write back custom data values in real time for consumption by other Deltix components and models. That opens up a huge range of opportunities ranging from risk management to portfolio strategies to processing efficiency. For example, if a particular CPU-intensive algorithm is used by multiple models, where's the sense in each model calculating those algorithm values individually and wasting CPU cycles? One component or trading model can calculate the values and write them back as a custom data stream for wider consumption - calculate once, use many.

Apart from viewing and editing data points, TimeBase Administrator also includes a charting component. Figure 2b shows a tick chart for Apple with the price ticks denoted by the plus marks (including one rather prominent bad tick highlighted by the red square) and the bid offer spread by the pale blue area.

One thing several of the Wrecking Crew remarked upon was the speed with which large data blocks loaded in charts from within the TimeBase Administrator. Then we noticed the small item in the message bar shown in Figure 2c...

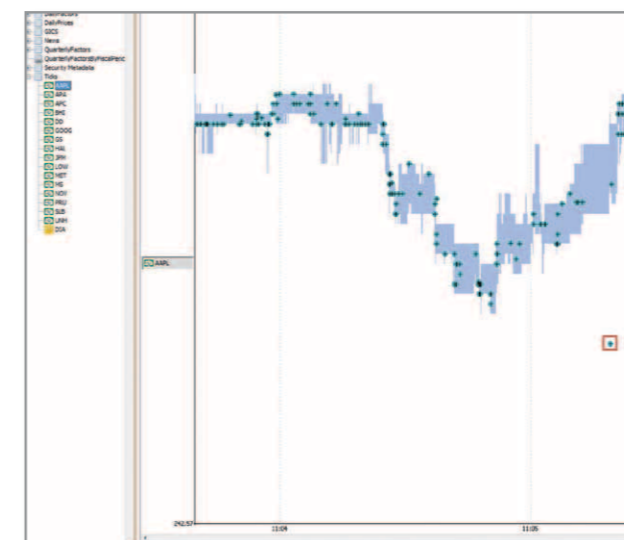


Figure 2b

Avg. 232,523 messages loaded in avg. 0.093739 seconds (2,480,545 messages per second)

Figure 2c

**Building models**

QuantOffice provides three methods for building trading models:

- **Visual Alpha (currently in beta):** a GUI-driven approach which allows the user to build models from either built-in or user-defined indicators and back test them.
- **Wizard:** a template-based approach where pre-built or user-built templates are used as the starting point for model development.
- **“Roll your own”:** strategies built manually using C# based upon existing code segments, or from scratch.

The three options are intended to cover various levels of user experience/expertise. Visual Alpha is a good starting point for those new to the Deltix platform and requires no direct interaction with underlying C# code. The wizard based approach minimises the amount of coding, though some tweaking will be necessary - especially if the user wishes to exploit more advanced trading concepts. The “roll your own” approach is really only suitable for experienced programmers/quants who are comfortable working directly with C# code in the .NET environment.

```
CodeTemplate + - * / == > < >= <= () AND OR
96 public void OnInit()
97 {
98
99 // Creating and initializing MesaSinewave indicator
100 mesaSinewave14 = new QuantOffice.FinancialAnalysis.TA.MesaSinewave(30
```

Figure 3a

**Visual Alpha**

We gave Visual Alpha (see Figure 3b) a try out by building a very basic strategy based upon John Ehlers's MESA (Maximum Entropy Spectral Analysis) sine wave indicator. Though - as Figure 3b shows - we managed a very modest degree of profitability (but only by ignoring costs), the key point was what was going on behind the scenes. Clicking Detailed Run or Generate in Visual Alpha automatically generates all the necessary C# files for the strategy to be run in real time through UHF Trader. Figure 3a shows a fragment of the generated code with the length parameter of the MESA sine wave indicator (which we had changed from its default of 14 to 30 in the Visual Alpha indicator dialog) highlighted.

While Visual Alpha's GUI doesn't offer the flexibility and power of the wizard or “roll your own” approaches, it is a very quick route to productivity. It's also intuitive; we couldn't find its documentation to start with, but this proved no obstacle to putting



Figure 3b

together and generating a strategy. It is also (as is the case with the strategy wizard) an extremely handy way to get up to speed with the mechanics of tweaking and writing C# model code:

- Build a model with Visual Alpha
- Examine the generated code
- Tinker with some parameters
- Re-examine the generated code
- Repeat

**Wizard**

To test the Deltix strategy wizard, we built an extremely crude (and barely profitable) model based upon the RSI. Opening the wizard from the main QuantOffice window allows you to define individual default elements of the strategy from among nine main component groups (see Figure 4 where we have opted to set the default data subscription as 27 second time bars built from tick data). In order to visualise any indicators that you plan to include in a model, click Chart Setup from the Indicators section of the wizard. Figure 5a shows the chart dialog with our default 27 second bars automatically pre-populated, along with the dialog

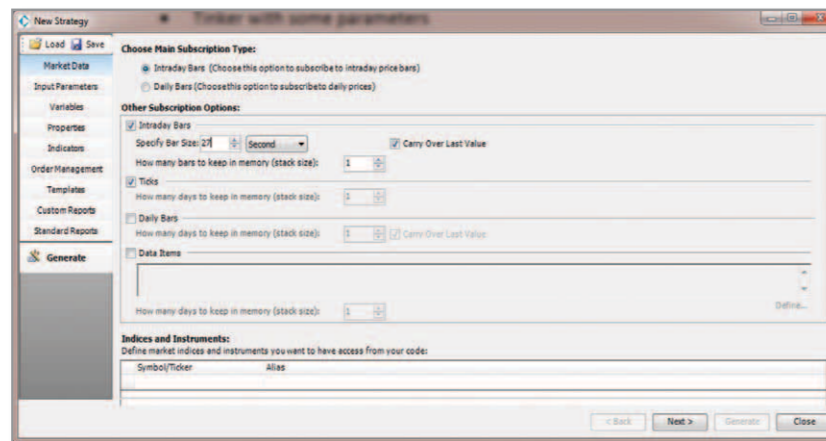


Figure 4

for adding a list of instruments - in this case our “portfolio” consisting of AAPL, GOOG and JPM.

In addition to limit/market/stop (we selected market), the Order section of the wizard includes a selection of simple execution algo strategies, which can of course be extended. From the Template section we selected one of the prebuilt strategy templates, plus Total PNL Intraday portfolio accounting (see highlighted areas in Figure 5b).

The strategy templates shown can of course also be extended by editing or supplemented by strategy templates built from scratch. (However, as will

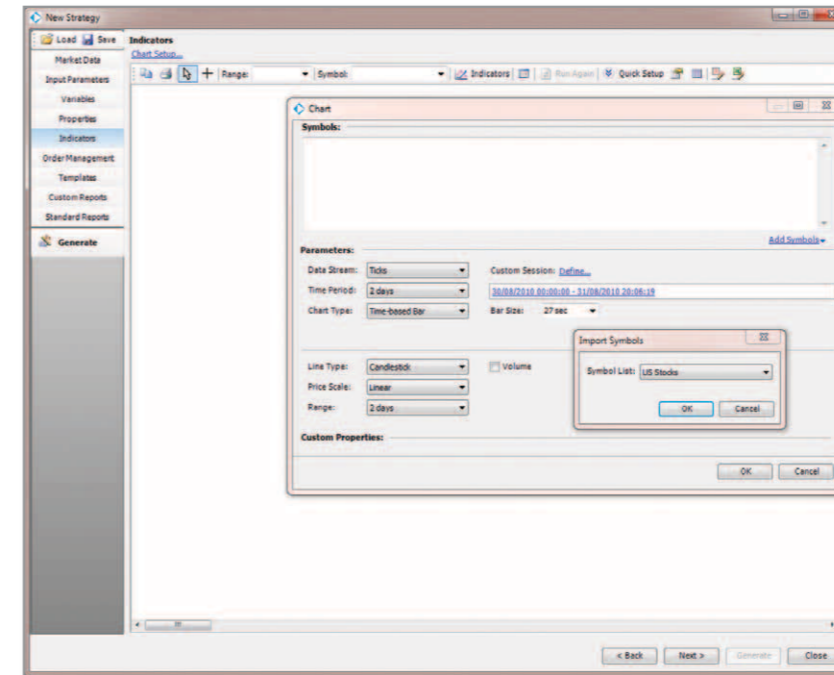


Figure 5a

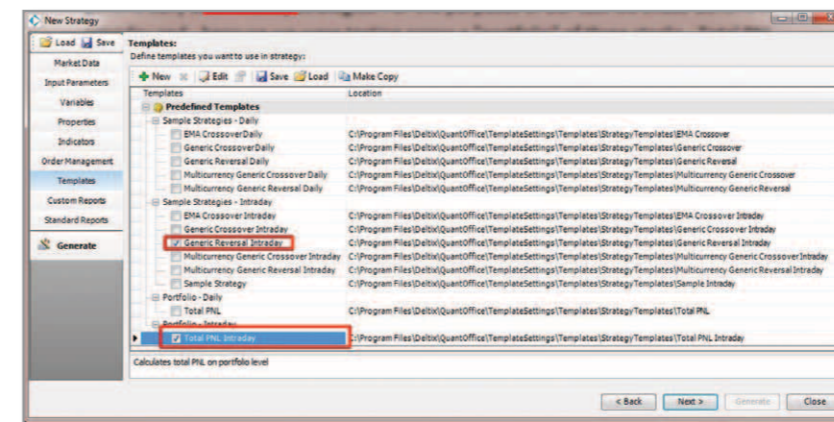


Figure 5b

become apparent later, the majority of users will probably opt to edit than build anew.)

After clicking next in the final step of wizard you are prompted to name your strategy and then auto generate its C# code. On completion you are asked if you wish to edit your strategy. Clicking OK brings up the strategy config window, which in this case also shows the list of five project files automatically generated. Clicking on any of these file names opens the associated code in a new Designer tab. (Files can also be opened in Microsoft VS (Visual Studio) and Strategy Designer can be set up as plug in to VS if desired). Most users will probably spend the majority of their time tinkering with the InstrumentExecutor.cs file, as this is where the main business logic of the model is contained.

However, this is not to ignore the importance of the

other files, which contain essential code infrastructure that auto generation spares you from having to create and debug from scratch - a MAJOR time saver (hence our earlier assertion that many users will probably choose to tweak existing template code rather than write code from scratch). Nevertheless, the fact that these “infrastructure” files are accessible means that advanced users or those with unusual requirements can make any desired adjustments.

Figure 6a shows the auto generated code for our RSI model, which will need some adjustment before any trades are triggered. The block of text framed in blue creates the basic variables for the strategy’s trigger lines, but initialises them with NaNs, so as they stand they won’t actually do anything. A further point here is that the default reversal template is based on the assumption that something like Bollinger Bands or Keltner Channels are being used and so placeholder variables suitable for those indicators have been created.

Therefore, some alternatives will need to be created and surplus variables deleted. A similar position applies in the case of the code framed in red in that the default syntax is intended for a “band based” reversal strategy. Again, some tweaking is necessary. (A more suitable strategy template for the RSI is actually available, but we wanted a pretext for more extensive code butchery to serve as an example - hence the less than perfect template choice.)

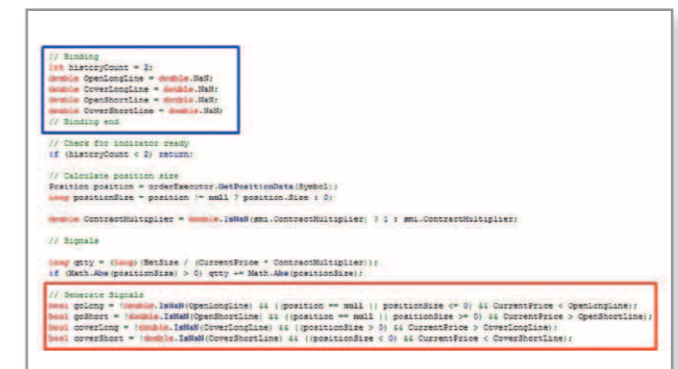


Figure 6a

```

// Binding
int HistoryCount = 2;
double RSLine = rsi.RSI;
double CoverEitherLine = 50;

// Binding end

// Check for indicator ready
if (HistoryCount < 2) return;

// Calculate position size
Position position = orderExecutor.GetPositionData(symbol);
long positionSize = position != null ? position.Size : 0;
double ContractMultiplier = double.IsNaN(position.ContractMultiplier) ? 1 : position.ContractMultiplier;

// Signals
long qty = (int)(Math.Ceiling((CurrentPrice - ContractMultiplier) / Math.Abs(positionSize)));

// Generate Signals
bool goLong = !double.IsNaN(RSLine) && ((position == null || positionSize <= 0) && RSLine > 20 && rsi.Previous < 20);
bool goShort = !double.IsNaN(RSLine) && ((position == null || positionSize >= 0) && RSLine < 80 && rsi.Previous > 80);
bool coverLong = !double.IsNaN(CoverEitherLine) && ((positionSize > 0) && RSLine > CoverEitherLine);
bool coverShort = !double.IsNaN(CoverEitherLine) && ((positionSize < 0) && RSLine < CoverEitherLine);
    
```

Figure 6b

Our final version of these two code segments can be seen in Figure 6b. Frankly, it's pretty horrible - but it works. Far more elegant ways of doing this are possible; for example, we haven't bothered creating variables for the trigger thresholds 20 and 80 that could then be used in parameter optimisation. Figure 7 shows a chart for the only profitable stock for our strategy with a clip of the overall strategy results overlaid.

**So?**

Probably the simplest way of giving a yes/no verdict on QuantOffice is to say that it was the closest I've come to having to break up a fight among the Wrecking Crew over possession of the mouse. So I suppose that's an emphatic "yes"...

On this occasion the Wrecking Crew ranged from arcade traders to a quant who can pretty much only communicate in C#. Despite that, QuantOffice enjoyed across the board approbation, which given the way it can be configured to scale from one user to hundreds is, to say the least, convenient.

On the matter of productivity, it delivers - period. As each member of the Wrecking Crew delved into the



Figure 7

# Barely scratching the surface

In giving our verdict on QuantOffice, it should be made clear that we have attempted to cover a product in six pages that probably requires at least sixty - and even that would omit all its associated components. Some of the myriad features we lacked the space to cover include:

- Importing models built in MATLAB
- The optimisation methods available – brute-force, genetic and dynamic (walk-forward) methods
- Custom output streams
- Scaling for rankings
- Creating additional variables/properties/input parameters in the strategy wizard
- Code locking
- Diagrammatic drag n'drop programming (though see Figure 8 for a consolation screenshot)

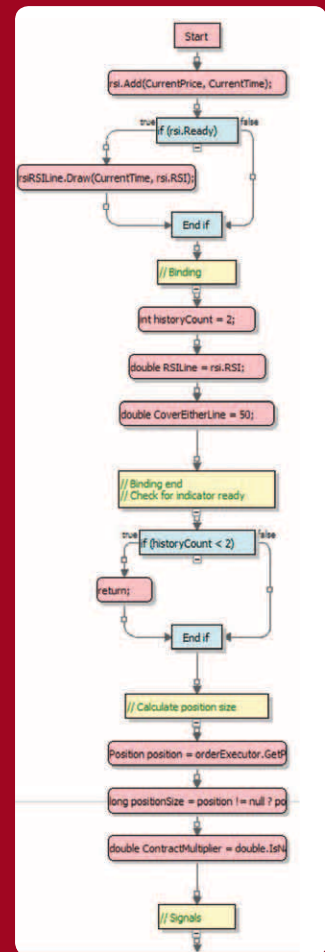


Figure 8

strategy development approach most appropriate to their experience/expertise, the lack of profanity and the speed of progress were striking.

In short, if you want rapid and flexible model development and deployment, you're in the right place - as will we be after our little chat with the Editor...

