

Pragmatix Trade Router

Version 1.6 Release Notes

12 August 2008

Introduction

This document describes the changes and improvements in the Pragmatix Trade Router software with respect to the previous version, 1.5. The Trade Router version number is shown in the application's startup screen. Alternatively, you can see the version number in the Help/About window.

New Features

Synthetic Stops

Some brokers do not support the 'stop' order type. This would mean that in the Trade Router, you can't protect your position with stop-loss and trailing stop orders. In order to allow for this kind of protection, *synthetic stops* have been implemented in the Trade Router.

With a synthetic stop, no stop order is sent to the exchange, but a trigger is created inside the Trade Router. The trigger fires as soon as the market price for the instrument reaches (or exceeds) the trigger level (which is, in effect, the stop price). When the trigger fires, the position is closed immediately with a market order.

Synthetic stops are used instead of real stops when the connectFIX.*.config file indicates that stop orders are not supported:

```
<!-- Does broker stop orders?, default="yes" -->  
<add key="SupportsStopOrders" value="no" />
```

The trigger is created as a virtual stop order inside the Trade Router. When the trigger fires, the virtual stop order is cancelled and a new 'Pos' order is created to close the position at market. Using a virtual order allows the Trade Router to administer the stop in the order database – just like all other orders.

Notes on Synthetic Stops

- In order for the trigger to fire, a good quality price data feed is crucial. Without the data feed, the trigger will not fire. Also, if the internet connection or the Trade Router itself fail, the market order to close the position will not be sent out, possibly resulting in significant losses.
- The three-phase entry/exit strategies can also use stop orders. We plan to offer synthetic stops here as well, in a future release of the Trade Router.

New data connection supported: ESignal

ESignal provides good quality market data via the internet against reasonable prices. In order to use ESignal's data in the Trade Router, you will need the following:

- An ESignal subscription, including the 'Desktop API' service;
- The ESignal application, installed on the Trade Router PC;
- A connectESignal.*.config file in the Trade Router 'config' folder.

ESignal must be running on the PC for the Trade Router to receive data from it. If it isn't running however, the Trade Router will start it up automatically.

Here is an example of a connectESignal.*.config file:

```
<?xml version="1.0" encoding="utf-8"?>
<settings>
  <add key="ApplicationString" value="YourUserName" />
  <add key="AutoReconnect" value="true" />
  <add key="UsesDynamicSubscriptions" value="false" />
</settings>
```

The only setting that must be set in order to connect to ESignal is `ApplicationString`; put in your ESignal user name (case sensitive).

In order to verify that you're entitled to use the Desktop API, ensure the Trade Router is running. In the ESignal window, go to menu 'Help'/'About', and click 'ActiveX info'. A window will open showing the current status.

Creating Price Data Engines for ESignal

In order to create a Price Data Engine for ESignal, you will need to indicate the ESignal symbol code. These symbol codes usually consist of two parts: the code for the symbol itself, a dash, and then the exchange code. For example, Volkswagen on Xetra is VOW-XET. For many American instruments, the exchange code is omitted, like 'MSFT' for Microsoft on the Nasdaq.

In the Trade Router, separate the symbol and the exchange code, like in the example below:

The screenshot shows a dialog box titled "Create new Engine" with the following fields and values:

- Engine name:** VW_Data
- Symbol:** VOW
- Exchange:** XET
- Instrument type:** Stock
- Option type:** (empty)
- Units per contract:** 1
- Currency:** EUR
- Minimal price variation:** 0,001

At the bottom, there is an "Example" section with an "ESignal symbol code" field and "Build" and "Test" buttons. The dialog box also has "OK" and "Cancel" buttons at the very bottom.

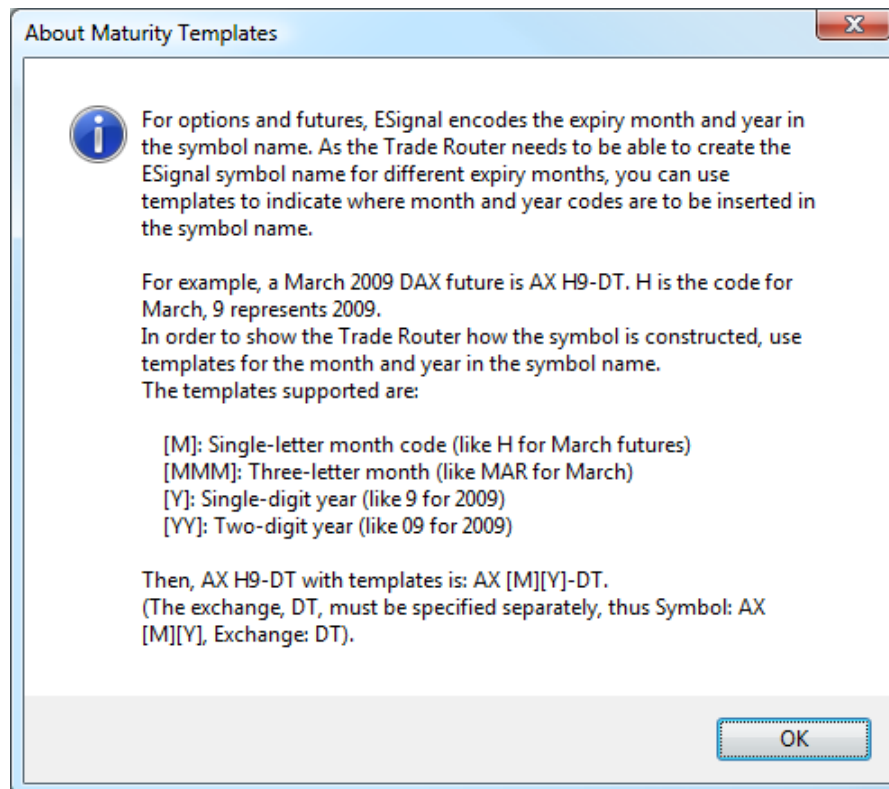
In the 'Example' section in the bottom part of the window, you can test the validity of the codes you've entered. Press 'Build' to have the application assemble the ESignal code from the information you supplied. Press 'Test' to request ESignal to validate the code for you.

Maturities

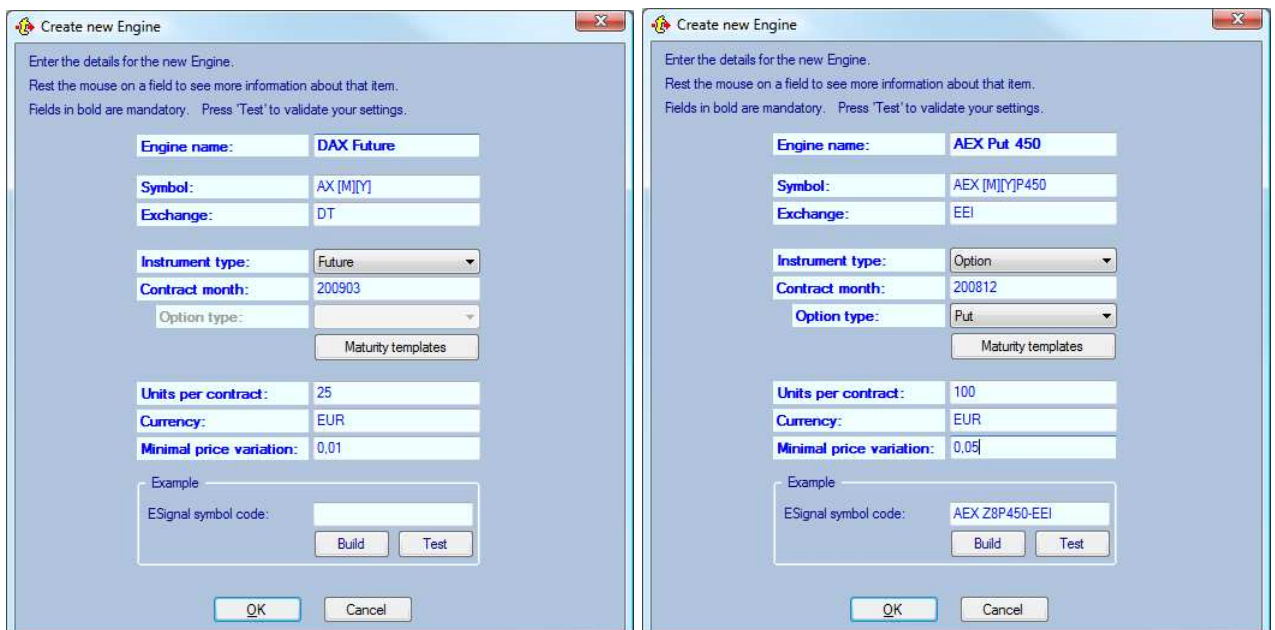
Expiration dates, strike prices and option types are all encoded within the ESignal symbol code. Unfortunately, the way the symbol code is formed is irregular.

For example, a DAX Sep09 future is coded as 'AX U9-DT', where AX stands for DAX, U9 for September 2009 and DT for DTB (the exchange). For an option contract, coding is more complicated, like 'AEX Z8P450-EEI' for a December 2008 AEX put option with a strike price of 450.00.

The Trade Router needs to be able to insert the maturity (expiration date) in the symbol code. In order to resolve this issue, *templates* have been created which can be placed in the symbol code and which indicate where the month and year codes are to be inserted. When you press the 'Maturities' button in the window shown above, a help window opens with additional information:



In order to define the DAX future and the AEX option mentioned earlier, you would fill in the fields as follows:



Dynamic Subscriptions

If you need to receive streaming prices inside the Trade Router for more than about 50 to 100 instruments, you may run into a few practical problems. First, your data provider may limit the amount of instruments simultaneously subscribed to. Second, when trading is intense (when the market opens, for example) you may run into performance problems due to the sheer amount of information received and processed in the Trade Router.

In case you only need these prices for synthetic or trailing stops, there is a simple solution. You can configure the Trade Router in such a way that streaming prices are only requested for an engine if that engine is in a position – after all, when there's no position, you don't need to set any stops.

This type of temporary data streams is achieved with *dynamic subscriptions*. You can declare, via the connect*.*.config file a connection to use this kind of subscriptions:

```
<add key="UsesDynamicSubscriptions" value="true" />
```

If you use dynamic subscriptions for Price Data Engines, a subscription for Price Data Engine X will start as soon as a Trade Engine receiving data from X enters a position. You can monitor the subscribed/unsubscribed status for Price Data Engines in the 'Remarks' column of the Price Data Engine list; if the stream is unsubscribed, you will see '(Stopped)' appear in that column. When the stream is subscribed, the column is empty, and you can see the prices come in.

Bug fixes

IG Markets FIX Execution Report: SellShort instead of Sell

Under certain circumstances, IG can confirm a Sell order with an execution report marked 'SellShort' instead of 'Sell', leading to incorrect interpretation of the report. This has been corrected, 'SellShort' is now interpreted correctly.

Create new Engine window

This window would not always close after the Engine had been created. This has been corrected.

InsideTradingWindow

During Trade Router startup, an exception may occur in the InsideTradingWindow routine. The exception is only visible in the application log, and unharmed in nature. Release 1.6 has fixed this issue.

Duplicate Engine names allowed when editing Engine settings

When editing settings of an existing Trade Engine, it is possible to specify a new name for that Engine. No 'duplicate name' check was performed there, leaving the possibility open to change the name to the name of an already existing Engine, leading to unexpected results. This has been fixed – the duplicate name check has been implemented.