FIX PROTOCOL
INDUSTRY-DRIVEN MESSAGING STANDARD℠

FIX Adapted for STreaming - FAST℠ Protocol Technical Overview

Written by Kevin Houstoun, Consultant HSBC, FPL Global Technical Committee Co-Chair

Table of Contents

Executive Summary 1
Background 1
What is FIX? 1
What is FAST℠? 2
Spiralling Market Data Volumes – A business problem 2
Solution Overview 3
FAST℠ – A technical solution 4
The reference code 6
Summary 6
References 6

Executive Summary

FIX Protocol Limited (FPL), the custodians of the FIX Protocol, have launched a new protocol called FAST℠, FIX Adapted for STreaming. FIX messages, like any self describing message syntax, have a relatively high overhead of message descriptor. The FAST℠ Protocol is a way of eliminating this overhead by exchanging the message description separately from the message. This, along with a number of other data compression techniques, allows FAST℠ to address the explosion in market data volumes and the technical challenges of transmitting them.

FIX Protocol Limited developed the FAST℠ Protocol by empirically testing a number of different compression techniques on market data from a number of the worlds leading exchanges in a sponsored proof of concept. The sponsors were Archipelago Exchange, Chicago Mercantile Exchange, International Securities Exchange, London Stock Exchange, Microsoft and Singapore Stock Exchange.

The FAST℠ Protocol has been demonstrated to be a highly efficient solution to the problem of spiralling market data volumes and is already being adopted by a number of the worlds leading exchanges. FIX Protocol Limited, drawing on previous experience of managing an open protocol, is publishing a reference implementation in a number of languages: C, C# and Java.

Background

What is FIX?

The Financial Information eXchange protocol was invented by Salomon Brothers and Fidelity Investments in the early 1990's to address the problem of communicating securities orders and their executions in the wholesale banking industry. Since the early 90's FIX has been expanded, through 7 protocol releases to cover all asset classes and extend its reach from

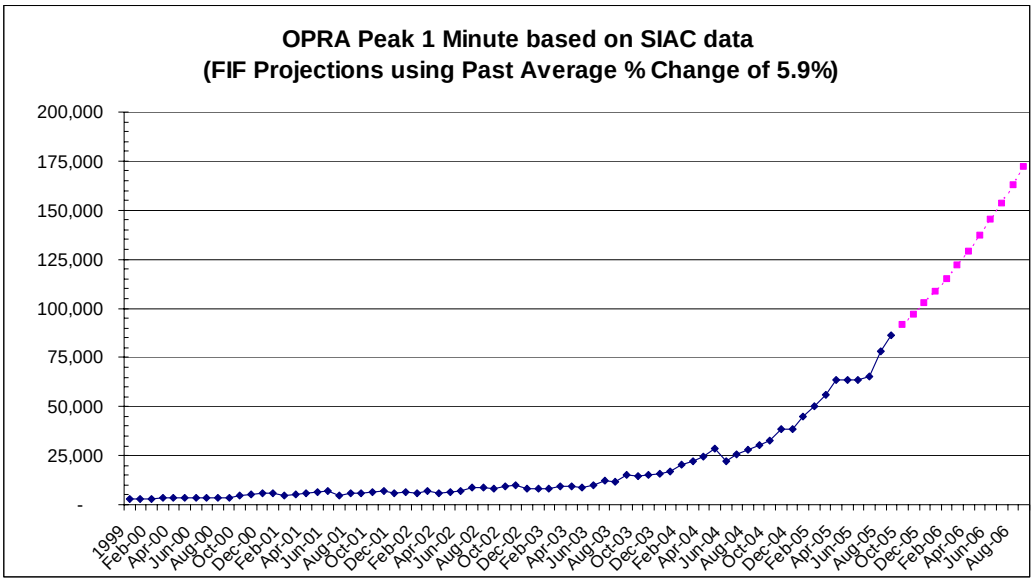
trade initiation to trade confirmation. FIX is today the most widely adopted protocol in the wholesale markets and the defacto standard for all institutional trading globally.

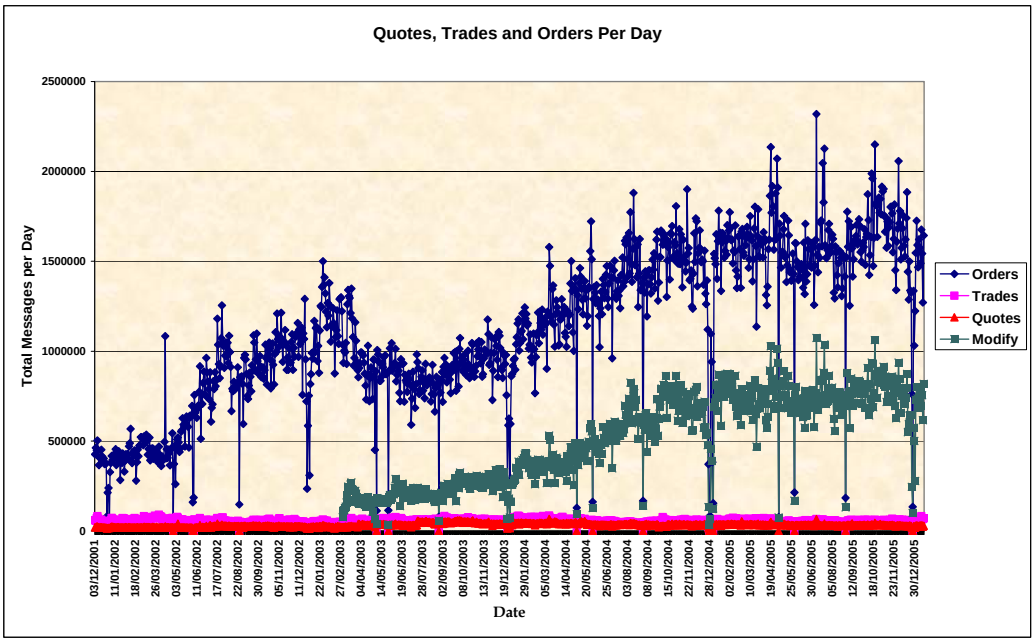
What is FAST℠?

By around 2003 FIX had reached such a position in the wholesale markets that it was becoming a victim of its own success. The ease with which market participants, money managers, investment banks, hedge funds, exchanges and matching mechanisms could connect was leading to a spiral in market data volumes as ever increasing numbers of electronic orders begat ever increasing volumes of market data.

Spiralling Market Data Volumes – A business problem

This trend is evident in the following illustrations. The first shows the volume of market data messages in the OPRA (Option Price Reporting Authority) consolidated feed, which represents all US derivatives business. The second shows the London Stock Exchange traded volumes in one of the worlds most advanced equity platforms. For more details see London Stock Exchange case study.





The increasing volumes of market data were causing delays, preventing market data from reaching traders in a timely fashion, thus disrupting their ability to trade. This was because ever increasing amounts of data were being pumped down the existing connections. In the short term this was addressed by exchange clients buying more lines. In the longer term a number of the worlds leading exchanges started discussing what could be done and contemplated rolling out a new series of proprietary exchange market data mechanisms.

To address these problems the FIX Protocol Global Technical Committee set up the Market Data Optimisation Working group with the following mission:

"To define standard services for the reliable transport of FIX market data messages using implicit tagging and support for multicast and point to point topologies"

This mission was to be fulfilled by approaching the problem with some basic aims:

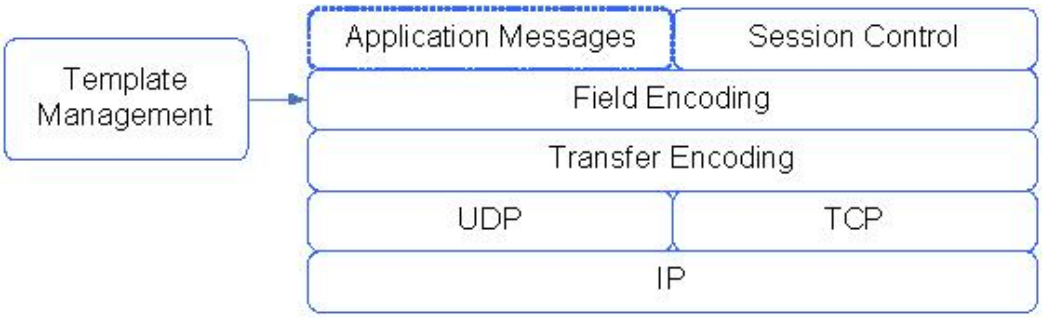
- Consider additional optimization techniques that may further enhance market data delivery such as compression
- Garner participation and representation from all industry segments in order to develop a true standards-based solution
- Create a common, flexible, straightforward solution that will reduce integration costs
- Make allowance for non-market data related information to take advantage of the same optimizations

Solution Overview

In 2005 FPL performed a proof of concept around these techniques that demonstrated that the FAST℠ Protocol could deliver significant bandwidth savings with reductions in market data latency and the decision was made to launch the new protocol.

FAST℠ – a technical solution

Broadly the design of the final FAST℠ Protocol is a series of layers as shown in the diagram below:



Application messages are the market data that need to be transmitted. Market data messages are handed into a Field Encoding layer that removes redundant data before they are handed off to the Transfer Encoding layer. The Transfer Encoding layer applies a number of techniques to compress the data into a minimum number of bytes. Once compressed the data is transmitted either using UDP in a multicast environment or TCP in a peer to peer session based protocol. Session Control governs such aspects of the communication as initiating and terminating FAST sessions, and error reporting.

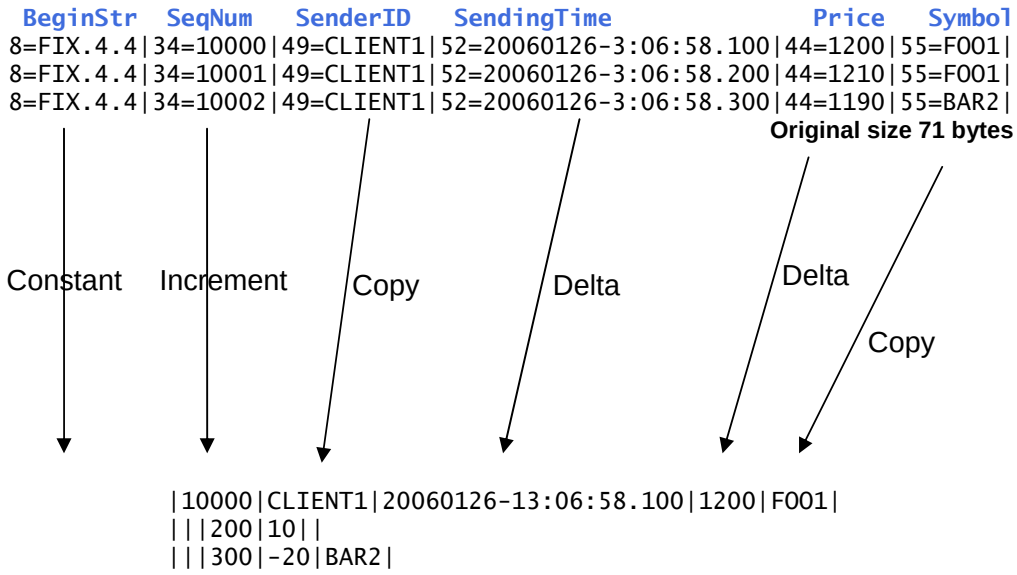
The Field Encoding layer removes redundant information by assigning multiple messages to the same logical frame, called a data stream in FAST℠, and using a template that describes the data format and optimisations. This allows FAST℠ to exploit similarities between consecutive messages to remove the need to transmit all the data content of a message.

The relationships are “Copy” when the value is a copy of the data transmitted in the previous message, “Increment” when you simply increment the value from the previous message, “Delta” when only the difference is sent, “Default where a default value is described in the template and the value is only sent if different” and “None” when there is no relationship and all the data on each message is sent. Message Templates describe the layout of each message type and define which field encoding technique is to be used on each field. A final type of encoding supported by the message template is “Constant” and this represents a field that always takes a particular value. These constant values are always simply added back into each message on decoding.

This message template is described in an XML template

```
<template name="ExampleOrder">
   <messageRef name="NewOrderSingle"/>
   <string name="BeginStr"> <constant value="FIX.4.4"/> </string>
   <u32 name="SeqNum"> <increment/> </u32>
   <string name="SenderID"> <copy/> </string>
   <string name="SendingTime"> <delta/> </string>
   <decimal name="Price"> <delta/> </decimal>
   <string name="Symbol"> <copy/> </string>
</template>
```

The effect of coding a message according to the rules described in the message template can be seen on the extract of three messages. The diagram below illustrate this



The Transfer Encoding layer compresses the data into the minimum physical space required on the wire. This compression is achieved utilizing a number of techniques.

In traditional FIX messages each field takes the form “Tag=Value<SOH>” where the tag is a number representing which field is being transmitted and the value is the actual data content. The ascii <SOH> character is used as a byte delimiter to terminate the field.

The first area of redundancy is the opening of every field with “Tag=”. These are mostly 2, 3 or 4 digit tags so they represent 3, 4 or 5 bytes of data that can be eliminated by exchanging a template that describes the message structure. This technique is known as implicit tagging as the tags become implicit in the data.

At the data field level a stop bit is used instead of FIX’s traditional <SOH> separator byte. Thus 7 bits of each byte are used to transmit data and the eighth bit is used to indicate if the end of the field has been reached. For market data where the average field length is approximately 2 bytes this saves approximate 33% of the bandwidth.

At the message level the first bytes of each message are a presence map indicating which fields are present in this particular message. If a field is shown as being transmitted in the presence map the transmitted field values then follow the map. If no fields are shown as present, all fields are derived from previous messages and the next byte is the start of the presence map for the next message.

Another technique, called binary encoding is used on number, this basically renders the number into binary across the 7 data bits in each byte. Thus a number less than 2^7-1, (127) will only occupy one byte, a number between 2^7 and 2^7*2 – 1 (16,383), will occupy two bytes etc. This saves space on many numbers, for example 16 represented as ASCII requires two bytes, binary encoded it only requires one.

The diagram below illustrates the effect of transfer encoding

```
 BeginStr  SeqNum   SenderID   SendingTime              Price   Symbol
8=FIX.4.4|34=10000|49=CLIENT1|52=20060126-3:06:58.100|44=1200|55=FOO1|
8=FIX.4.4|34=10001|49=CLIENT1|52=20060126-3:06:58.200|44=1210|55=FOO1|
8=FIX.4.4|34=10002|49=CLIENT1|52=20060126-3:06:58.300|44=1190|55=BAR2|
```

Original size 71 bytes

```
FIX.4.4|10000|CLIENT1|20060126-13:06:58.100|1200|FOO1|
FIX.4.4|10001|CLIENT1|20060126-13:06:58.200|1210|FOO1|
FIX.4.4|10002|CLIENT1|20060126-13:06:58.300|1190|BAR2|
```

Implicit Tagging
54 bytes (-24%)

```
FIX.4.41000OCLIENT120060126-13:06:58.10O120OFOO1
FIX.4.41001CLIENT120060126-13:06:58.20O121OFOO1
FIX.4.41002CLIENT120060126-13:06:58.30O119OBAR2
```

SBIT Encoding
48 bytes (-33%)

```
FIX.4.4nnCLIENT120060126-13:06:58.10OnnFOO1
FIX.4.4nnCLIENT120060126-13:06:58.20OnnFOO1
FIX.4.4nnCLIENT120060126-13:06:58.30OnnBAR2
```

SBIT + Binary Encoding
43 bytes (-39%)

The reference code

FPL had experience of launching a protocol before and felt that one of the problems with any protocol specification was that it was open to interpretation and that any ambiguity is both difficult to spot and interpreted in at least two different ways. Indeed a large part of the maturing process of FPL has been the clarification of those ambiguities. To avoid this pitfall in the FAST℠ Protocol FPL decided to publish a reference implementation. The prototype has been developed in C and it was anticipated that most implementations would be in C due to its speed.

Summary

FPL has launched a new protocol called FAST℠. The FAST℠ Protocol is designed to stream financial market data using as little bandwidth and introducing as little processing latency as possible.

References

This whitepaper is largely based on material presented at a series of FAST briefings by FIX Protocol Limited. The original material can be found on the http://www.fixprotocol.org/fast