COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

# The FIX Protocol

Compiled and edited

for CFRM 520 (002c)

# References


- Presentation by Brian Driscoll, *Introduction To The Financial Information eXchange (FIX) Protocol*, 2008 (https://www.slideshare.net/BrianDriscoll/barcampnyc3-intro-to-fix)

- Presentation by Dr Richard Motie, *An Introduction to the FIX Protocol*, 2013
- (https://www.slideshare.net/richardmotie/fix-protocol-an-introduction-r-motie)

- FIX Trading Community (http://www.fixtradingcommunity.org/http://www.fixtradingcommunity.org)

# What is the FIX Protocol?

- It is a messaging standard for the real time electronic exchange of securities transaction data.

# Who uses FIX?

- Institutional investors (the buy side)
- Broker/dealers (the sell side)
- Exchanges & ECNs
- Financial industry utilities
- Software & services vendors

# What is it used for?

- It's used by exchanges, ECNs, & brokers/ dealers to distribute market data, quotes, etc.
- Money mangers use it to send orders and receive executions from brokers.
- It's used by exchanges & ECNs to receive orders or quotes & report trades.
- It's used to allocate & confirm trades.
- These are only a few examples.

# Products & regions supported

- Financial Products Supported
  - Equities
  - Fixed Income
  - Derivatives
  - FX

- Used worldwide
  - Americas
  - UK
  - Europe
  - Asia
  - Pacific Rim

# How is the standard governed?

- It's a public domain standard
- Owned & managed by FIX Protocol, Ltd
- It's a non-profit financial community organization
- Couple hundred member firms
- Volunteers from member firms & industry work on the specification
- Official website: http://www.fixtradingcommunity.org

# A brief history of FIX

- 1992 1st used between Fidelity & Salomon
- Jan 1995 - FIX 2.7
  - Public spec released
  - Now obsolete
- Dec 1995 – FIX 3.0
  - Now obsolete
- 1996 – FIX 4.0
  - Good US equity support
  - Still in use in US
- 1998 – FIX 4.1
  - Incremental release
- 2000 – FIX 4.2
  - Better international
  - Equity market data & allocations
  - Futures, options, FX
- 2001 – FIX 4.3
  - Fixed income, XML
- 2003 – FIX 4.4
  - Confirms & trade reporting
- 2006 – FIX 5.0 & T1.1
  - Complex FX, improved session/transport level

# Growth of FIX

- Began as IOI (Indication of Interest) - Bid/Offers.
- Moved to Orders and Executions:
- Actual Orders to buy and sell shares in securities, and responses from the counterparty about how those Orders were executed.
- FIX has become the most commonly used protocol for electronic financial transactions.
- Used by – ECNs, Brokers, Portfolio Managers, Exchanges.
- Covers major asset classes – Equities, Derivatives, FX, Fixed Income.
- Supports Pre-Trade, Trade, and Post-Trade messaging.

# FIX: General Overview

- A standard way to communicate trading information electronically between brokers, buy-side institutions, and markets.

- An industry driven messaging standard for the exchange of trading related information between financial institutions.
  - Platform independent.
  - Flexible.
  - Open (free) protocol

# What is a FIX message?

- FIX message contains three parts:
  - Header
  - Body
  - Trailer

- FIX message is a collection of fields eg Symbol, Order ID, Price, Order Quantity

- Each field is a tag-value pair: <tag>=<value>

- Two types of FIX messages:
  - Application Level
  - Session Level

# The three parts of a FIX message

- Header – contains administrative information about the message – who sent the message, who the message is going to, what time it was sent.

- Body – contains the actual financial information – fields like Symbol, Shares, Price.

- Trailer – usually there is only one field set in here – the Checksum for the message – to ensure message integrity.

# Message types

- Both *session level* and *application level* messages are constructed the same way (tag/value pairs), but the functionality of the two types is different.
- An application level message contains the actual financial data that a user will care about.
- Application level messages include Orders, Execution Reports, Indications of Interest, Order Cancels, Allocations.
- Session level messages are used for administrative functionality, and connectivity between FIX engines,
- Session level messages include – FIX Login messages (establish a connection to a remote FIX engine), FIX Logoff messages, FIX Resend Request messages.

# Message types (cont'd)

- Session level msgs
  - Logon
  - Heartbeat
  - Test Request
  - Resend Request
  - Reject
  - Sequence Reset
  - Logout
- Application level msgs
  - 100+ msg types
  - New order – single
  - Execution report
  - Order cancel/replace request
  - Order cancel request
  - Allocation
  - Trade Capture Report
  - Confirmation

# What is meant by a tag value pair?

- Tag:
  - FIX uses predefined Tags.
  - Each Tag represents a specific Field.
  - Each Tag is given a predefined number.
  - FIX dictionary provides the list of Fields and corresponding Tag numbers.
- Value:
  - Values represent the value the Tag is assigned to.
  - Supported data types are: int, float, quantity, Price, char, Boolean (Y/N), Currency, timestamp, string.
- For example: Symbol is tag 55. If there were a FIX order for Intel, it would be:

**55=INTC**

# Message Header: All administrative fields

- Who sent the message
- To whom the message was sent
- Who was the trader that sent the message (if applicable)
- What version of FIX is being used in the message
- What type of message it is
- Where the message is to be routed to (if applicable)

# Header Field Examples

- **8=FIX 5.0** Message is using Version 5.0 of FIX protocol (BeginString)
- **52=20090724-13:40:24** What time the message was sent (GMT) (SendingTime)
- **49=Fidelity** Fidelity is sending the message, specified in the header using the SenderCompID field
- **56=ML** Message being sent to Merrill Lynch, specified in the TargetCompID field
- **35=D** Fidelity sending Merrill Lynch an Order, specified in the MsgType field
- **D** signifies an Order. Other values for MsgType include:
  - **8** (Execution Report)
  - **6** (Indication of Interest)
  - **A** (FIX logon message)

# Message Body

- The Body of a FIX message contains the actual financial information that will be important to the receiver of the message.

- Important fields that could be in the body of a FIX Order message are:
  - Symbol (tag 55)
  - Order Quantity (tag 38)
  - Order Type (tag 40)
  - Price (tag 44)
  - Side (tag 54)
  - Time In Force (tag 59)

# Message Body Example

- Create an order to buy 10000 shares of Yahoo at the Market, *Good Until Cancelled*

  - OrderQty – 10000: **`38=10000`**
  - Symbol – Yahoo: **`55=YHOO`**
  - Order Type – Market Order: **`40=1`** (1 indicates Market Order)
  - Time In Force – Good until cancelled: **`59=1`** (1 indicates GTC*)
  - Side – buy: **`54=1`** (1 indicates buy)

*Good (un)Til Canceled

# Message Trailer

- The Trailer of a FIX message usually contains only one field – tag 10 the CheckSum.

- The Checksum in a FIX message provides a simple message integrity check. The Checksum is the sum of all the *binary* values in the message, eg **`10=193`**

- More specifically:
  - The checksum algorithm of FIX consists of summing up the decimal value of the ASCII representation all the bytes up to the checksum field (which is last) and return the value modulo 256. (http://fix-protocol.blogspot.com/p/hello-guys-hope-you-must-be-doing-well.html)

- Upon receiving a message, a FIX engine will compute the checksum of the message, and make sure it matches the checksum in the actual message, this helps to identify transmission problems.

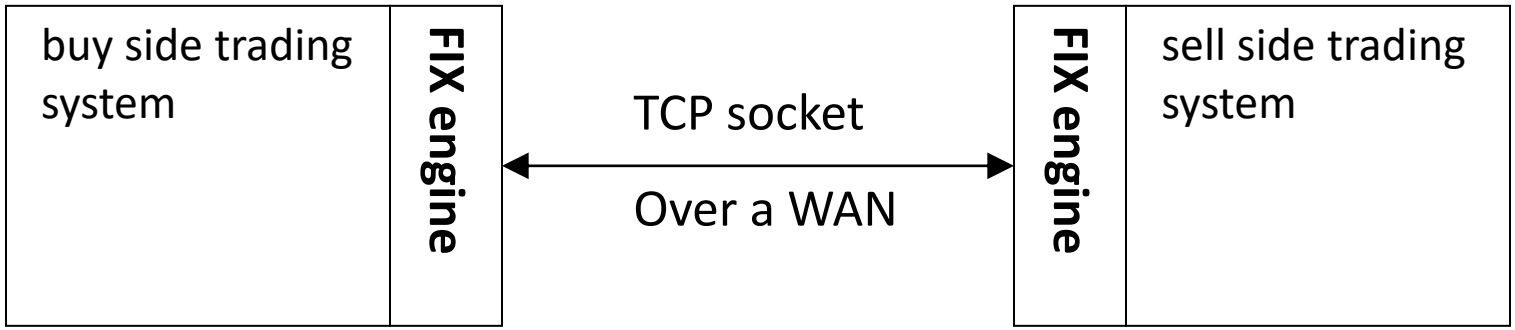
# Two message syntaxes

- Tag=Value syntax
  - 4 components to each field
  - <Tag>=<Value><Delimiter>
    - <Tag> is the tag number of the field
    - <Value> is the value of the field
    - <Delimiter> is an ASCII control character ( ^, | )

- FIXML syntax
  - XML schema defined
  - Application messages only, no session level
  - Tag=Value msgs can contain FIXML in payload
  - Not used as much as Tag format, in practice

# A simple example scenario

**A money manager sends an order to a broker, and receives an execution/fill back**




FIX Session Layer Transport via TCP used with tag=value message syntax where the buy side initiates connection and the sell side accepts.

# What is a socket?

- A socket is an "endpoint for network communications." In other words, it's the virtual device that your program uses to communicate with the network
- When you want to write bytes out over the network, you write them to a socket
- When you read from the network, you read from that socket
- In this way, it is similar to the way a "file" is used to interact with hard drive storage

- Taken from https://www.scottklement.com/rpg/socktut/introduction.html

# A socket is comprised of

- An IP address
- A transport protocol (eg TCP, UDP)
- A port number
- Taken from http://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket

# The difference between a socket and a port

- A socket
  - represents a single connection between two network applications
  - For trading, this means two different computers
  - Sockets are bidirectional, meaning that either side of the connection is capable of both sending and receiving data.
- Taken from http://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket

# The difference between a socket and a port

- A port
  - represents an endpoint or "channel" for network communications
  - Port numbers allow *different applications on the same computer* to utilize network resources without interfering with each other
  - Port numbers most commonly appear in network programming, particularly socket programming
  - Normally, a Web site uses port number 80 (by default) and this number need not be included with the URL (although it can be)
  - In IP networking, port numbers can theoretically range from 0 to 65535. Most popular network applications, though, use port numbers at the low end of the range (such as 80 for HTTP)

- Taken from http://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket

# Typical simplified message flow

- Buy side connects TCP socket to predefined port on sell side FIX Engine
- Sell side accepts TCP connection
- Buy side sends a Logon msg
- Sell side sends a Logon msg back
- Buy side sends New Order – Single msg
- Sell side sends Execution Report acknowledging order
- Sell side sends Execution Report containing fill

# Buy 5000 IBM @ 110.75

**8=FIX.4.2^9=251^35=D^49=AFUNDMGR^56=ABROKER^34=2^**
**52=20030615-01:14:49^11=12345^21=1^ 55=IBM^54=1^**
**60=2003061501:14:49^38=5000^40=2^44=110.75^10=127**

**Header fields:**

8=BeginString (indicates FIX 4.2)

9=BodyLength

35=MsgType (new order)

49=SenderCompID (AFUNDMGR)

56=TargetCompID (ABROKER)

34=MsgSeqNum (2)

52=SendTime

**Trailer Fields:**

10=Checksum

**Body fields:**

11=ClOrderID (client order id)

21=HandleInst (automated exec)

55=Symbol (IBM)

54=Side (buy)

56=TransactTime

38=OrderQty (5000)

40=OrdType (Limit)

44=Price (110.75)

52=SendTime

# Same message in FIXML

```
<FIXML>
<FIXMLMessage>
  <Header>
    <SendingTime>20030615-
01:14:49</SendingTime>
    <Sender>
       <CompID>AFUNDMGR
       </CompID>
    </Sender>
    <Target>
      <CompID>ABROKER
      </CompID>
    </Target>
  </Header>
```

```
<ApplicationMessage>
  <Order>
    <ClOrdID>12345
    </ClOrdID>
    <HandlInst Value="1"/>
    <Instrument>
      <Symbol>IBM
      </Symbol>
    </Instrument>
    <Side Value="1"/>
    <TransactTime>
     2003061501:14:49
    </TransactTime>
```

```
    <OrderQtyData>
        <OrderQty>5000
        </OrderQty>
    </OrderQtyData>
    <OrdType Value="2"/>

    <Price>110..75</Price>
    </Order>
  </ApplicationMessage>
</FIXMLMessage>
</FIXML>
```

# Integrating applications w/ FIX

- Write your own FIX Engine
  - Complex, time consuming, etc
  - Appropriate in some situations
- Use an existing FIX Engine
  - Many existing FIX Engines
  - Most platforms and languages supported
  - Integrate it into the application

# Open source FIX Engines

- QuickFIX
  - C++, Python, & Ruby APIs
  - Windows, Linux, Solaris, Mac OS X
  - www.quickfixengine.org

- QuickFIX/J
  - is a 100% Java implementation
  - www.quickfixj.org

- QuickFIX/N
  - Native .NET implementation
  - http://quickfixn.org/

# FIX engine vendors

- Many FIX engine vendors (~20)

- Some of the major vendors are:
  - OnixS
    - http://www.onixs.biz/cpp-fix-engine.html
  - B2Bits
    - http://www.b2bits.com/trading_solutions/fix_engines.html
  - NYFIX
    - https://www.ullink.com/buy-side/#nyfixapi

# FIX Info on the Web

- http://www.fixtradingcommunity.org

  - The web site for FIX Protocol Ltd
    - Specifications
    - Documentation
    - Discussion forums
    - Formerly www.fixprotocol.org

- Much of the info in this presentation came from this web site

# FAST Protocol

- **F**IX **A**dapted for **ST**reaming

- Technology standard developed by FIX Protocol Ltd, specifically aimed at optimizing data representation on the network

- It is used to support high-throughput, low latency data communications between financial institutions

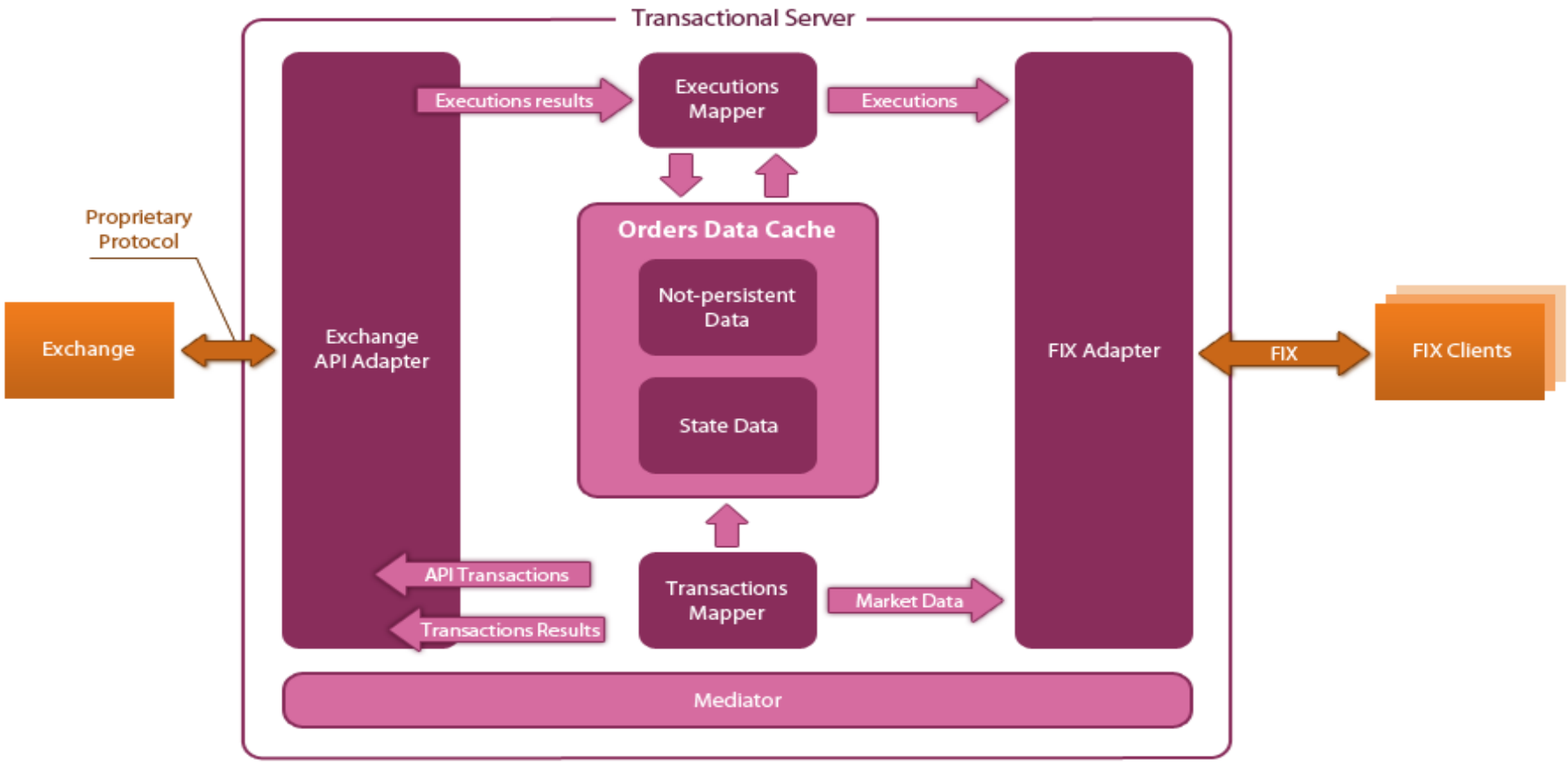
- In particular, it is a technology standard that offers significant compression capabilities for the transport of high-volume market data feeds and ultra low latency applications

- https://en.wikipedia.org/wiki/FAST_protocol

# FAST Protocol

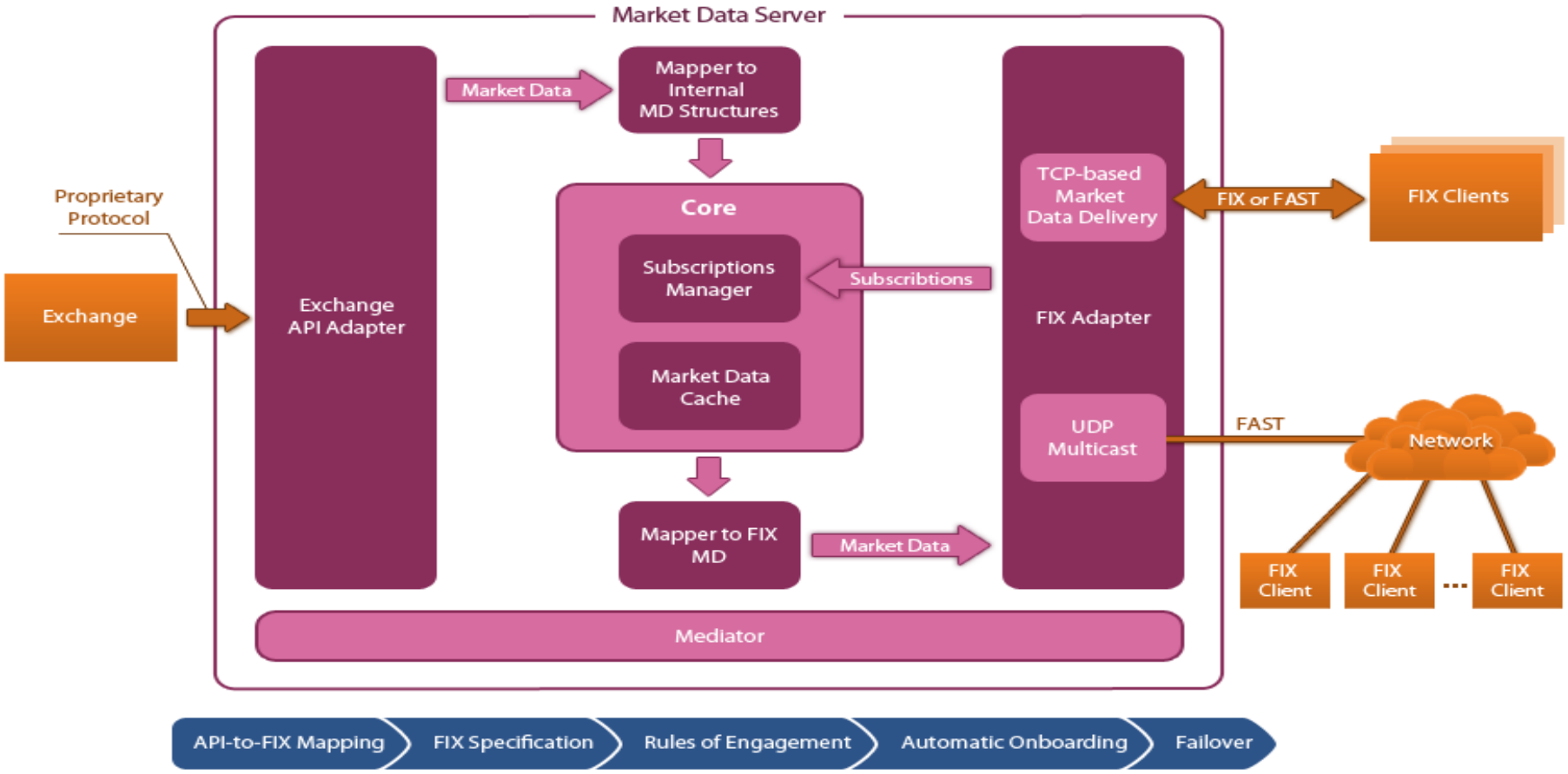
- The original purpose of FAST was optimization of FIX messages
- The encoding method has been generalized to apply to a wider set of protocols (eg UDP, as well as TCP)
- The protocol is intended to enable efficient use of band width in high volume messaging without incurring material processing overhead or latency
- The protocol was developed as a result of the Market Data Optimization Working Group's Proof of Concept (POC) project to explore various methods of optimized data representation within the FIX Protocol
- http://www.fixtradingcommunity.org/pg/structure/tech-specs/fast-protocol

# Putting it all together (trades)




http://www.b2bits.com/consulting/fix_trading_clients/fix_to_exchange_framework.html

# Putting it all together (mkt data)




http://www.b2bits.com/consulting/fix_trading_clients/fix_to_exchange_framework.html

**[End]**