NYSE Technologies℠

Powering the exchanging world.℠

Learn to Speak FIX

NYSE Technologies' FIX Training Class

Training Agenda

9:30 AM – 10:00 AM	FIX – An Overview
10:00 AM – 10:15 AM	FIX Versions & Asset Classes
10:15 AM – 10:45 AM	FIX in Detail
10:45 AM – 11:00 AM	The FIX Engine
11:00 AM – 12:00 PM	FIX Session Layer
12:00 PM – 1:00 PM	Lunch
1:00 PM – 2:30 PM	The FIX Application Layer
2:30 PM – 2:45 PM	FIX Routing
2:45 PM – 3:15 PM	Integrating FIX into an Electronic Trading Infrastructure
3:15 PM – 3:45 PM	Rules of Engagement
3:45 PM – 4:15 PM	What's New in FIX?
4:15 PM – 4:30 PM	Case Studies
4:30 PM – 5:00 PM	Q&A

NYSE Technologies℠

Powering the exchanging world.℠

Module I: An Overview of FIX

What is FIX?

Stands for Financial Information eXchange

- Is a de-facto standard to communicate trading information electronically between brokers, buy-side institutions, and markets
- Is a flexible method of handling various types of financial information
- Is platform independent - works with various types of computer and communication systems (i.e. X.25, TCP/IP, copper, fiber, satellite, etc.)
- Is industry-driven, open and free protocol that can be used by anybody, to talk to anybody else who is also using FIX

The FIX Protocol website (www.fixprotocol.org) is the central point of reference and discussion

Who Uses FIX?

Buy-side:

- Banks
- Investment Managers / Financial Advisors
- Portfolio / Fund Managers

Sell-side:

- Broker-Dealers (Market Makers, Agents, Inter-dealer Brokers)
- Exchanges
- ECNs / ATSs (light, dark pools)

Vendors

- OMS / EMS
- Front-office / back-office
- FIX Order Routing Networks

What Asset Classes are covered?

- Equities
- Derivatives – i.e. Futures & Options
- Fixed Income – i.e. Bonds, Munis, Repos
- Foreign Exchange – i.e. swaps, forwards
- Collective Investment Vehicles (CIV) – i.e. Mutual Funds, ETFs, Unit Trusts, Managed Investments

What message types are covered?

Pre-trade

- Offerings, IOI’s, Trade Advertisements
- Quotes, Market Data

Order and trade

- Single/Multi-orders
- Executions

Post-trade

- Allocations, Confirmations, Affirmations
- Trade / Position Reports
- Settlement Instructions

Why was FIX created?

Historically traders use to deal over phone – trading instructions were given by phone, and information on fills was relayed back to the buy-side by phone as well.

- Inefficient and slow
- Error-prone
- Dealers had to sit by phone all day

In 1992, Fidelity and Salomon created a set of rules and defined different kinds of messages to exchange electronically so that they exchange financial information in an efficient way

The growth of FIX through the ages

FIX began with Indications of Interest

- Indication of Interest (IOI) simply a message to a counterparty that you are interested in buying or selling a certain security
- IOIs are generally very high volume – on any given day a large institution might send out tens of thousands of IOIs
- This was easily automated through FIX

Enhanced to support Orders and Executions

- Actual orders instructions to buy and sell securities
- Responses from the counterparty about how those orders were executed

The growth of FIX through the ages

- New message types – Quotes, Allocations, News, Email, etc.

- New functionality to existing messages – support more products (futures, options, fixed income, forex) and for program trading

- Polishing for global use
  - Sending of raw (binary) data in messages to support non-English characters
  - Providing for support of 24 hour a day / 7 day a week trading
  - Adapting for local trading requirements (different types of symbols)

FIX Today

- Has become the most commonly used standard for electronic financial transactions
- Used by ECNs, Brokers, Portfolio Managers, Exchanges
- Many exchanges support FIX or derivatives of FIX
- Used in the Americas, Europe, the Middle East, Africa, Asia, and Australia

Who governs FIX?

FIX is overseen by FIX Protocol Limited (“FPL”)

- Organization of industry participants using FIX who oversee all modifications to the FIX protocol
- Allows users to make suggestions for enhancements to the protocol (new fields, new messages)

FPL mission

- “To improve the global trading process by defining, managing, and promoting an open protocol for real-time, electronic communication between industry participants, while complementing industry standards.”
- Business Working Groups & Technical Working Groups drive new initiatives forward

Why use FIX ?

Business Reasons

- **Accommodates higher volumes** with fewer personnel
- **Widely adopted** – can leverage the active participation of industry experts via working groups
- Prepares firms for **shortened settlement cycles**
- Designed to **achieve Straight Through Processing** (STP)
- Reduces data copy errors and risk
- **Promotes liquidity** through IOIs
- Responds quickly to industry changes
- Traders can spend more time "*working*" an order

Why use FIX ?

Technical Reasons

- Delivers information in **real time**
- Provides platform and vendor **independence**
- **Eliminates proprietary interfaces** and coding of multiple message formats reducing *time to market*
- Guarantees **sequential message delivery**
- Supports **data security** (encryption)
- Supports multiple currencies and instrument types
- Encourages reusable infrastructure for more **cost-effective** connectivity

NYSE Technologies℠

Powering the exchanging world.℠

Module II: FIX Versions and Asset Classes

FIX Functionality Matrix

Message Support	FIX 4.0 [Jan 1996]	FIX 4.1 [Apr 1998]	FIX 4.2 [Mar 2000]	FIX 4.3 [Aug 2001]	FIX 4.4 [Apr 2003]	FIX 5.0 [Oct 2006]
Equities						
➢ Basic Order flow	🟩	🟩	🟩	🟩	🟩	🟩
➢ IOIs and Advertisements	🟩	🟩	🟩	🟩	🟩	🟩
➢ Quotes	🟩	🟩	🟩	🟩	🟩	🟩
➢ Market Data	🟥	🟥	🟩	🟩	🟩	🟩
➢ Allocations	🟨	🟨	🟩	🟩	🟩	🟩
➢ Confirms / Affirms	🟥	🟥	🟥	🟥	🟩	🟩
➢ Trade Reporting	🟥	🟥	🟥	🟨	🟩	🟩
➢ Program Trading	🟨	🟨	🟩	🟩	🟩	🟩
➢ Algorithmic Trading	🟨	🟨	🟨	🟨	🟩	🟩
Futures & Options						
➢ Basic Order flow	🟥	🟨	🟩	🟩	🟩	🟩
➢ Multi-leg Order flow	🟥	🟥	🟨	🟩	🟩	🟩
➢ IOIs and Advertisements	🟨	🟩	🟩	🟩	🟩	🟩
➢ Quotes	🟥	🟨	🟩	🟩	🟩	🟩
➢ Market Data	🟥	🟥	🟩	🟩	🟩	🟩
➢ Allocations	🟥	🟨	🟩	🟩	🟩	🟩
➢ Confirms / Affirms	🟥	🟥	🟥	🟥	🟩	🟩
➢ Trade Reporting	🟥	🟥	🟥	🟨	🟩	🟩
➢ Security and Position Reporting	🟥	🟥	🟥	🟥	🟩	🟩
➢ Collateral Management (Listed derivatives)	🟥	🟥	🟥	🟥	🟥	🟩

Legend	🟥	No Support	🟨	Some Support	🟩	Good Support	⬜	Not Applicable

NYSE Technologies

FIX Functionality Matrix

Fixed Income						
➢ Basic Order flow	🟥	🟥	🟨	🟩	🟩	🟩
➢ Multi-leg Order flow (Repos, swaps/switches/rolls)	🟥	🟥	🟥	🟨	🟩	🟩
➢ IOIs (offerings)	🟨	🟨	🟨	🟩	🟩	🟩
➢ Quotes	🟥	🟨	🟨	🟩	🟩	🟩
➢ Allocations	🟥	🟨	🟨	🟩	🟩	🟩
➢ Confirms / Affirms	🟥	🟥	🟥	🟥	🟩	🟩
➢ Trade Reporting	🟥	🟥	🟥	🟨	🟩	🟩
➢ Collateral Management	🟥	🟥	🟥	🟥	🟩	🟩
Foreign Exchange						
➢ Basic Order Flow (spots,forwards)	🟥	🟨	🟩	🟩	🟩	🟩
➢ Basic Order Flow (swaps)	🟥	🟨	🟨	🟨	🟨	🟩
➢ Quotes (spots, outright forwards, FX swaps)	🟥	🟨	🟩	🟩	🟩	🟩
➢ Market Data (executable streaming prices)	🟥	🟨	🟨	🟨	🟨	🟩
➢ Allocations	🟥	🟨	🟨	🟨	🟨	🟨
➢ Confirms / Affirms	🟥	🟥	🟥	🟥	🟨	🟨
➢ Trade Reporting	🟥	🟥	🟥	🟨	🟨	🟨
General						
➢ News	🟩	🟩	🟩	🟩	🟩	🟩
➢ Email	🟩	🟩	🟩	🟩	🟩	🟩
➢ Transport Independence Framework	⬜	⬜	⬜	⬜	⬜	🟩
➢ Regulatory Compliance	⬜	⬜	⬜	⬜	⬜	🟩

Legend	🟥	No Support	🟨	Some Support	🟩	Good Support	⬜	Not Applicable

Growth of FIX

FIX Version	3.0	4.0	4.1	4.2	4.3	4.4	5.0
Release Date	Sept 95	Jan 97	Apr 98	Mar 00	Apr 01	Apr 03	Dec 06
# Volumes	1	1	1	1	7	7	7
# Admin Msgs	7	7	7	7	7	7	7
# Business Msgs	17	20	21	39	61	86	94
# Fields	112	138	208	396	641	914	1139
# Appendices	4	4	7	16	>25	+	+
# Pages in spec	57	69	106	265	745	+	+

Importance of FIX Versions

- Version has **significant impact on application-level** business processing

- Which FIX messages and versions will you support is influenced by
  - Your trading application capabilities
  - Your counterparties' capabilities (and FIX versions)

FIX Version Usage

- Today the **most commonly used versions of FIX are 4.0 and 4.2**
- FIX 4.1 is rarely used
- FIX 4.3 is also rarely used. This version enhanced support for Options, Futures, and Mutual Funds
- **FIX 4.4** is preferred for **Fixed Income** products. It expanded functionality for Equities, Futures, Options, and Foreign Exchange
- FIX 5.0 is not widely used by anyone at the moment

Why would I use FIX 4.2?

- Most commonly used version of FIX
- **Market data** messages – allow for sending and receiving of ECN and exchange book data
- Mass quoting – bundle up multiple quotes in one message
- Provisions for **FIXML** – instead of using the FIX tag-value pair syntax, you can instead send messages via the FIXML format
- Also added messages types and fields to **support program and list trading**

Why would I use FIX 4.3?

- Enhanced support for **complete trade life cycle**
  - Pre-Trade, Trade, Post-Trade

- New message types to support **Quotation, Market Data, Security Definition, Cross** and **Multi-Leg Orders**

- **Trade reporting** – allowing sell-side firms to provide details of completed trades to an external entity not involved in execution of the trade

- Added Support of Collective Investment Vehicles (CIV) for **Mutual Funds**

Why would I use FIX 4.4?

- Improved functionality for **Fixed Income**
  - The FPL and the Bond Market Association worked hand-in-hand to make this the preferred FIX version for Fixed Income

- Enhanced **multi-leg order flow**

- Introduced **Collateral Management** messages

Why would I use FIX 4.4?

- Expanded functionality for **Equities and Foreign Exchange**
  - The capability to transmit **Confirmations and Affirmations** was introduced

- Expanded Functionality for **Futures and Options**
  - **Position Maintenance and Reporting** were introduced in FIX 4.4 to support listed derivatives clearing

Recommended Reading

- http://www.fixprotocol.org/specifications/
- FIX 4.0 Specification (“the basics”)
- FIX 4.2 Specification (“core”)
- FIX 4.2 Appendix D Order State Change Matrices
- FIX 5.0 Specification
- FIX Implementation Guide

Useful FIX Tools

- FIXionary (http://www.fixionary.com)

- FIX Specifications (http://www.fixprotocol.org/specifications/)

- FIXimate (http://fixprotocol.org/FIXimate3.0/)

NYSE Technologies℠

Powering the exchanging world.℠

Module III: FIX in Detail

What is a FIX message?

- It is a series of tag-value pairs

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

- There are two categories of FIX messages
  - Application
  - Session

- FIX Connection is comprised of three parts: log-on, message exchange, and logout

What do you mean by tag-value pair?

- The FIX protocol specifies hundreds of different “tags” or different fields that could be in a message

- In an actual FIX message, each tag/field name is represented by a **number and value**

- For example, if there was a FIX order for Intel, the symbol field would look like:

55=INTC

FIX specification defines each tag

Field ID (TAG)	Field Name	Format	Description
1	Account	String	Account mnemonic as agreed between broker and institution.
2	AdvId	String	Unique identifier of advertisement message. (Prior to FIX 4.1 this field was of type int)
3	AdvRefID	String	Reference identifier used with CANCEL and REPLACE transaction types. (Prior to FIX 4.1 this field was of type int)

4	AdvSide	Char	Broker's side of advertised trade
			
			Valid values:
			B = Buy
			S = Sell
			X = Cross
			T = Trade

FIX Tag Data Types

- **int**
- **float**
- **Qty**
- **Price**
- **char**
- **Boolean (Y/N)**
- **String**
- **MultipleValueString:** String field (see definition of “String” above) containing one or more space delimited values.
- **Currency**
- **Exchange:** String field (see definition of “String” above) representing a market or exchange.
- **UTCTimestamp:** YYYYMMDD-HH:MM:SS or YYYYMMDD-HH:MM:SS.sss

Delimiter for tag-value pairs

- A delimiter character known as the **SOH** (ASCII code 01) separates the different tag/value pairs to indicate the end of each field.

- It can look like or ^A or | in different log file representations.

Who is Who?

- **Initiator** establishes the telecommunications link and initiates the session via transmission of the initial Logon message.

- **Acceptor** is the receiving party of the FIX session. Performs first level authentication and declares the connection request “accepted” through transmission of an acknowledgment Logon message.

What are the parts of a FIX message?

- There are three parts – Header, Body, and Trailer
  - **Header** – contains administrative information about the message – who sent the message, whom the message is going to, what time it was sent
  - **Body** – contains the actual financial information – fields like Symbol, Order Quantity, Price
  - **Trailer** – usually contains only one field – the CheckSum for the message – to ensure message integrity

Sample Order Message - Header - Body - Trailer

Header fields - grey

```
8=FIX.4.29=013235=D57=Simulator
34=2 49=SLGM056=SBI0 52=20070
315-14:58:28 55=IBM 40=2 38=1000
21=2 11=order 0 60=20070315-
18:10:10 54=1 44=110.5 10=097
```

Trailer - grey

Body fields - black

Can you explain the Header further?

- The Header has all the administrative fields
  - **Who sent** the message
  - **To whom** the message was sent
  - Who was the **trader** that sent the message (if there was one)
  - What **version** of FIX is being used in this message
  - What **type** of message is it
  - Where the message is to be **routed to** (if applicable)

Mandatory FIX 4.2 Header Fields

Tag	*Field Name*	*Req'd*	*Comments*
8	BeginString	Y	Identifies beginning of new message and protocol version. ALWAYS FIRST FIELD IN MESSAGE. (Always unencrypted) Valid values: FIX.4.2
9	BodyLength	Y	Message length, in bytes, forward to the CheckSum field. ALWAYS SECOND FIELD IN MESSAGE. (Always unencrypted)
35	MsgType	Y	Defines message type. ALWAYS THIRD FIELD IN MESSAGE. (Always unencrypted). Note: A "U" as the first character in the MsgType field (i.e. U1, U2, etc) indicates that the message format is privately defined between the sender and receiver. Valid values: *** Note the use of lower case letters *** 0 = Heartbeat…………
49	SenderCompID	Y	Assigned value used to identify firm sending message.
56	TargetCompID	Y	Assigned value used to identify receiving firm
34	MsgSeqNum	Y	Integer message sequence number. *)*
52	SendingTime	Y	Time of message transmission (always expressed in UTC (Universal Time Coordinated, also known as “GMT”)

Can you explain the Header further?

- If Fidelity was sending a message, they would need to specify in the header that they are the sender using the **SenderCompID** field, which is tag **49** in FIX

  49=Fidelity

- If Fidelity was sending this message to Merrill Lynch, that would be specified in the **TargetCompID** field, which is tag **56**

  56=ML

Can you explain the Header further?

- If Fidelity was sending Merrill Lynch an **Order,** this would be specified with the **MsgType** field, which is tag 35

  35=D

- "D" signifies an Order. Some other possible values for MsgType can be
  - 8 (Execution Report)
  - 6 (Indication of Interest)
  - A (FIX Logon message)

Can you explain the Header further?

- If a trader at Fidelity by the name of Joe was sending ML an Order, this would be specified with the **SenderSubID** field (tag **50**)

50=JOE

- Conversely, there is also a **TargetSubID** field, if there was a specific trader at ML (let's call her Jane) that the Order was being sent to (tag **57**)

57=JANE

Can you explain the Header further?

Other important fields in the header:

- **BeginString** (tag **8**) – What version of FIX is this message
  - 8=FIX.4.0 (can also be FIX.4.2, FIX.4.4, etc.)
- **SendingTime** (tag **52**) – what time the message was sent (always in GMT)
  - 52=20070926-13:58:28

Can you explain the Body further?

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

- Other important fields that could be in the Body of an Order message are:
  - Symbol (55)
  - Order Quantity (38)
  - Order Type (40)
  - Price (44)
  - Side (54)

Can you explain the Body further?

- If I wanted to create an order to buy 8,000 shares of Yahoo at market price, and have it remain in force until I cancel it:
  - OrderQty – 8000

    38=8000
  - Symbol – YHOO

    55=YHOO
  - OrdType – Market

    40=1 (1 Indicates Market Order)
  - TimeInForce – Good Till Cancel

    59=1 (1 Indicates Good Till Cancel)

What does the Trailer do?

- The Trailer of a FIX message contains one field – **CheckSum (tag 10)**

- Simple message integrity check

- Formula based upon the number of characters in the FIX message resulting in a 3 digit number

10=193

- Upon receiving a message, a FIX engine will compute the checksum to make sure it matches the checksum value sent in the message

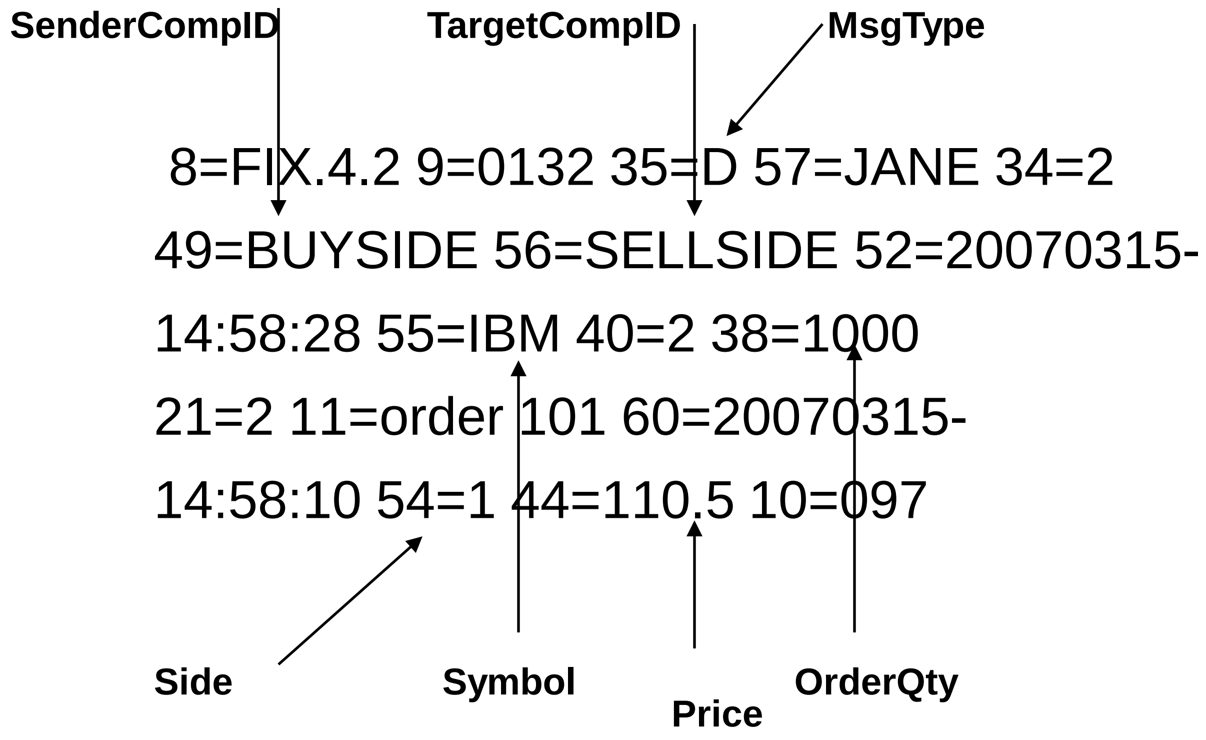
Is calculated by summing every byte of message up to *Checksum* field and transforming it into modulo 256 number.

- For example if byte sum of a FIX message is 514, value of Checksum would be *002*

Calculated after encryption if message being encrypted

A complete FIX message – New Order Single

BUYSIDE asks SELLSIDE to buy 1000 shares of IBM



```
 8=FIX.4.2 9=0132 35=D 57=JANE 34=2
49=BUYSIDE 56=SELLSIDE 52=20070315-
14:58:28 55=IBM 40=2 38=1000
21=2 11=order 101 60=20070315-
14:58:10 54=1 44=110.5 10=097
```

Categories of FIX messages

- There are two categories of FIX messages
  - **Application** messages
  - **Session** messages

- Both are constructed the same way (as a series of tag-value pairs), but functionality is different

What is an Application message?

An Application message is a message that contains actual financial information

Application messages include

- Orders
- Execution Reports
- Indications of Interests
- Order Cancel Requests
- Allocations

What is a Session message?

Session messages are used for **administrative** functionality and connectivity between FIX engines

Session messages include:

- Logon (establish a connection to a remote FIX engine)
- Logout
- Resend Request

The use of these messages will be detailed in the FIX Session Layer section

Field Constraints

- Concept of **Mandatory, Optional, and Conditionally Required** Fields
- Determined by type of message being sent
- Optional fields may become mandatory based on values of other fields
  - Price field is required if Order Type = Limit
- Conflicting fields may not be present at same time

Customization to FIX

- Trading parties agree on specific usage and content of fields subject to the guidelines and definitions provided in the specification

- FIX provides for custom fields and messages to support extensions or specific trading requirements

What else should I know about FIX?

- There are a few other concepts about FIX that get mentioned from time to time
  - FIX is an **optimistic** protocol
  - FIX is **point-to-point**

When someone says FIX is an *optimistic protocol* what do they mean?

- The term “optimistic protocol” means that if I have sent a FIX message, **I assume my counterparty has received it**
- FIX does not have “transactional delivery” – so, except for a few messages, **most FIX messages are unacknowledged**
- However, FIX does have an **error recovery mechanism** and a set of rules about resending missed messages

What does someone mean when they say FIX is *point-to-point*?

- In the context of FIX, **“point-to-point”** means that I must establish a FIX session with each of my counterparty and I must send my messages **directly** to them

- For example, if I was connected to **100 different counterparties** and wanted to send an Indication of Interest to each of them, I would have to send the same message **100 times**

Review

- Name two categories of FIX messages
  - Session, Application
- Name three parts of a FIX message
  - Header, Body, Trailer
- Fields such as Sender Company ID, Message Type, and Sending Time are located in which part of a FIX message?
  - Header
- A simple integrity check is included in which tag, that is also the final tag of every FIX message?
  - CheckSum (tag 10)

Review

- Are Header tags allowed within the Body of a FIX message?
  - No

- What, if anything, is wrong with this FIX message?

```
8=FIX.4.2 9=0216 50=Simulator 34=9 49=SBI0 56=SLGM0
52=20070906-15:19:40 151=0  55=MSFT 35=8 40=2
11=order1 31=110.5 150=2 39=2 54=1 44=110.5 32=1000
17=40775735 38=1000 21=2 60=20070906-15:19:40.593
 6=110.5 20=0 14=1000  37=40751938 10=192
```

  - The first three tags in a FIX message should always be 8, 9, and 35

Review

- What, if anything, is wrong with this FIX message?

  8=FIX.4.2 9=0132 35=D 57=Simulator 34=2 49=SLGM0 52=20070906-15:19:35 55=IBM 40=2 38=1000 21=2 11=order0 60=20070906-15:19:33 54=1 44=110.5 10=099

  - TargetCompID (tag 56) is missing

- What, if anything, is wrong with this FIX message?

  8=FIX.4.2 9=0133 35=D 57=Simulator 34=6 49=SLGM0 56=SBI0 52=20070906-15:19:35 55=EBAY 40=2 38=1000 21=2 11=order4 60=20070906-15:19:31 54=1 44=110.5

  - CheckSum (tag 10) is missing; it should be the last tag in each message

Review

- What, if anything, is wrong with this FIX message?

8=FIX.4.2 9=0132 35=D 57=Simulator 34=2 49=SLGM0 56=SBI0 52=20070906-15:19:35 55=IBM 40=2 38=1000 21=2 11=order0 60=20070906-15:19:31 54=1 44=110.5 10=099

Nothing

- What, if anything, is wrong with this FIX message?

8=FIX.4.2 34=11 9=0216 35=8 50=Simulator 49=SBI0 56=SLGM0 52=20070906-15:19:40 151=0 55=IBM 40=2 11=order0 31=110.5 150=2 39=2 54=1 44=110.5 32=1000 17=40775719 38=1000 21=2 60=20070906-15:19:40.640 6=110.5 20=0 14=1000 37=40751937 10=128

Tag 34 (MsgSeqNum) is in the wrong place. The first three fields in every FIX message should always be BeginString(8), BodyLength(9), and MsgType(35), in that order.

NYSE Technologies℠

Powering the exchanging world.℠

Module IV: The FIX Engine

What is a FIX Engine?

A FIX engine is a piece of software that manages:

- The “FIX session” (external network connection) to counterparties
  - Establishing, maintaining, and breaking **connections**
  - Automatic **Session-layer messaging**: Heartbeats; Test Requests; Logons; Logouts

- FIX message **formatting and validation** (rejection of invalid messages) both inbound and outbound

- Message-level **encryption** management

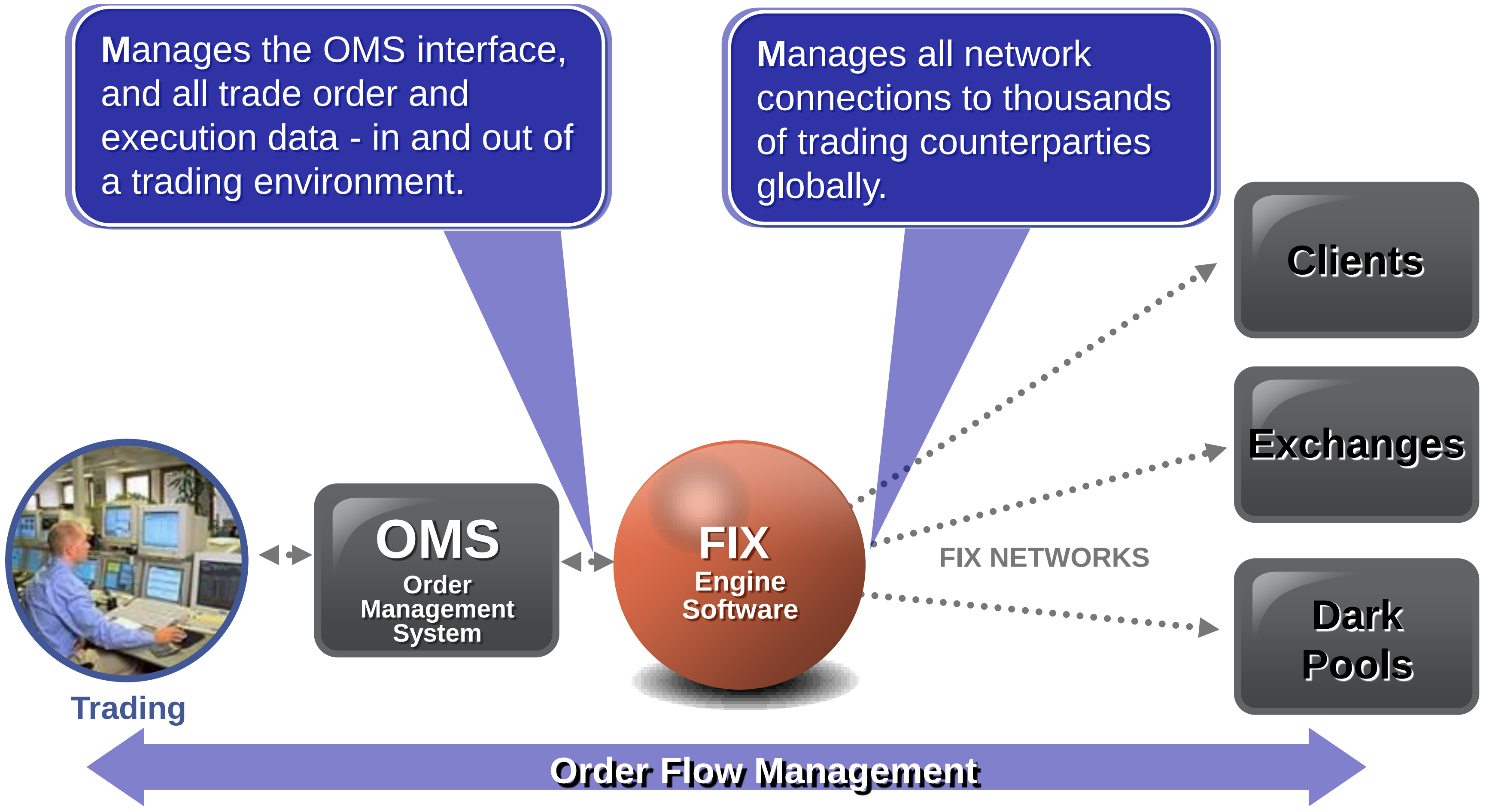
What is a FIX Engine?

A FIX engine is a piece of software that manages:

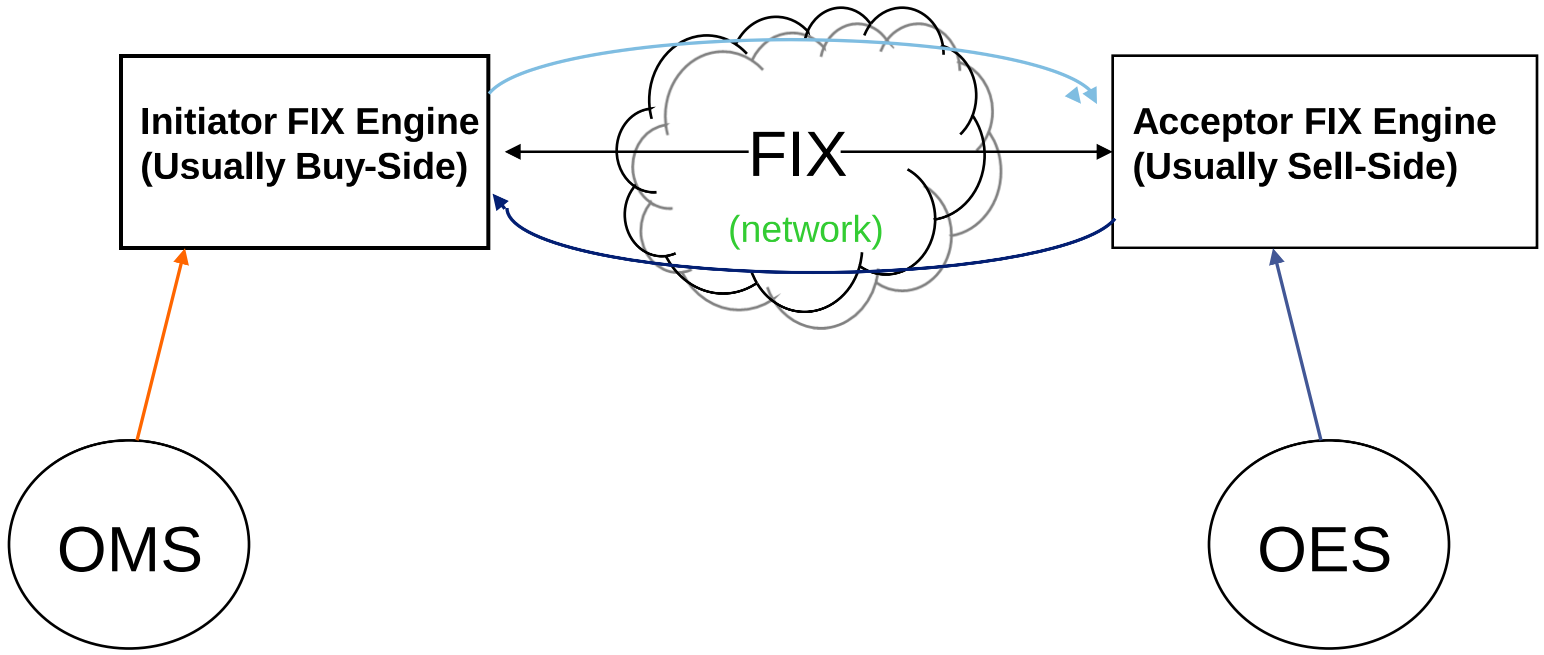
- Message **logging, persistence**, and gap recovery
  - Sequence number maintenance
  - Resend Requests and Gap Fills

- Propagating **application messages upstream / downstream** to the order/execution management system

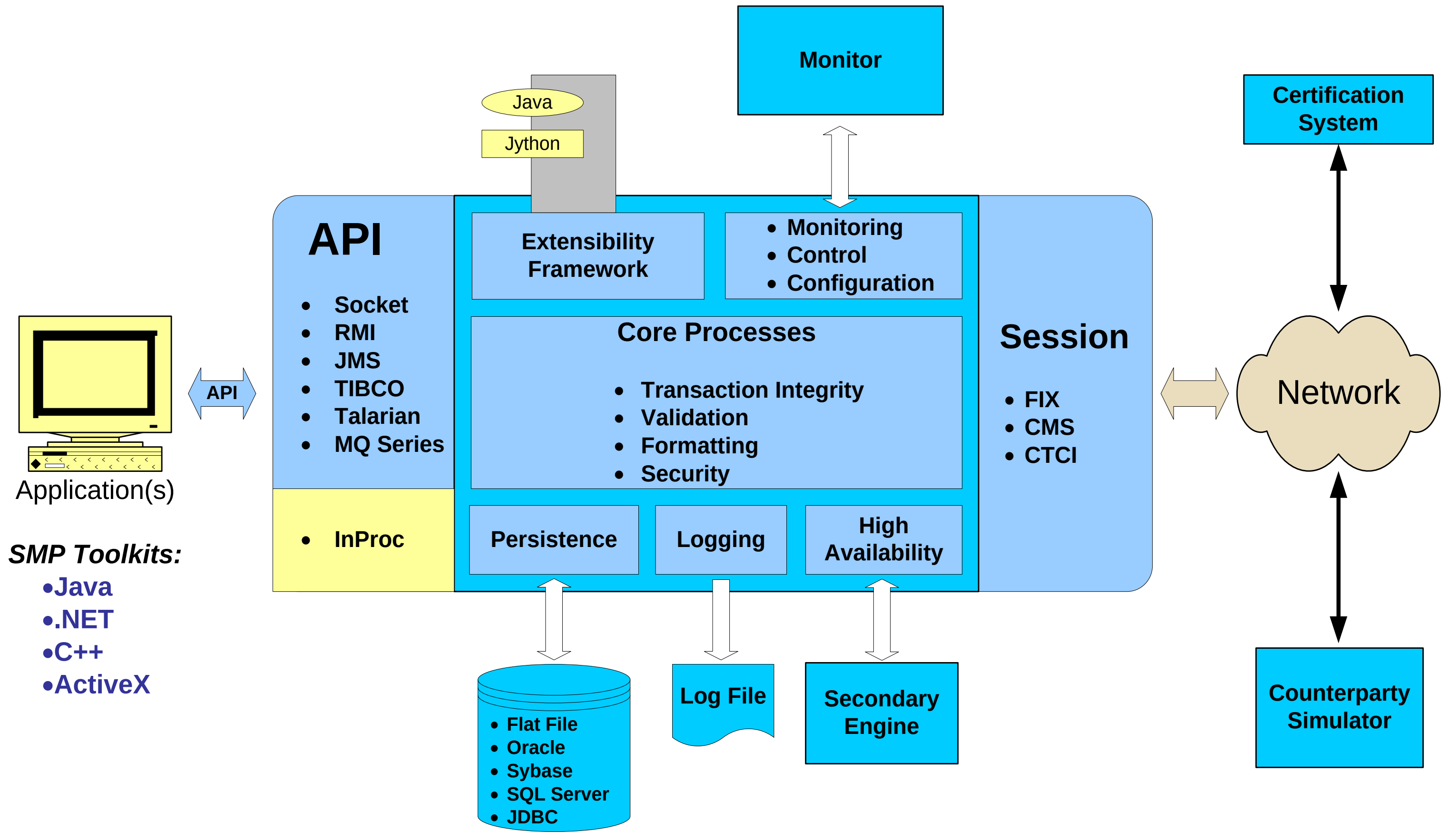
In the context of a trading system, your FIX **engine is the interface to the outside world**, which, together with a network, connects you to your counterparties and allows you to trade and exchange related information in a standard fashion.


Manages the OMS interface, and all trade order and execution data - in and out of a trading environment.
Manages all network connections to thousands of trading counterparties globally.
Trading
OMS
Order Management System
FIX
Engine Software
FIX NETWORKS
Clients
Exchanges
Dark Pools
Order Flow Management

FIX Engine Interaction Between Counterparties


Initiator FIX Engine (Usually Buy-Side)
FIX
(network)
Acceptor FIX Engine (Usually Sell-Side)
OMS
OES

What's Inside?



SMP Toolkits:
- Java
- .NET
- C++
- ActiveX

NYSE Technologies

What FIX Engine Features do you need?

- Good technical fit to organization – Operating System / API

- Scalability / Performance

- Flexibility / Configurability relative to
  - Development
  - Run-time operation
  - Support and monitoring

- Redundancy / Fail-over capability (High Availability)

- FIX versions support

What is a FIX Engine Not Supposed to Do?

- Not supposed to perform any **business logic**

- A basic FIX engine shouldn’t do the **work of an OMS:**
  - Decide when to **cancel an Order**
  - Automatically **send an Execution Report** in response to an Order
  - Reject an Order with a **duplicate Client Order ID**

What is a FIX Engine Not Supposed to Do?

- Should not be responsible for keeping order state
  - The FIX engine **shouldn’t keep track of the state of an order** (i.e. CumQty, AvgPx, LeavesQty calculations).
  - This is **typically performed by the OMS** or a routing application with a state manager

- It should **not do any automatic routing** unless specifically configured otherwise
  - The engine ideally provides all **Session-layer management** while acting as **pass-through for the Application layer** to the OMS and other trade applications

NYSE Technologies℠ Powering the exchanging world.℠

Module V: The FIX Session Layer

FIX Session Layer

- The FIX Session layer handles all the **administrative functionality** of the FIX protocol

- As with any protocol, not only is the information in the message important, but **how the message gets from one place to another**

What is the Session Layer?

A session layer of any protocol defines how two sides using that protocol communicate:

- How they know they are **logged on** (connected to each other)
- How they know when they are **logged off**
- How to determine the **order of the messages**
- How to **recover** if some messages are missed or duplicated

What is the Session Layer, anyway?

- Important concept – **need two FIX engines** to have a FIX session
- If you have only **one FIX engine** running, and it has no other FIX engine to talk to, then there is no FIX session!
- Much like you need **two people to have a conversation**

What type of messages are in a FIX Session Layer?

- Heartbeat (35=0)
- Logon (35=A)
- Test Request (35=1)
- Resend Request (35=2)
- Session-level Reject (35=3)
- Sequence Reset (35=4)
- Logout (35=5)

So how does it all get started?

- The Initiator (normally buy-side) will establish the telecommunications link and send a **FIX Logon message to the Acceptor** (normally sell-side)

- Upon receiving the Logon message, the acceptor **must acknowledge** that message with its own Logon

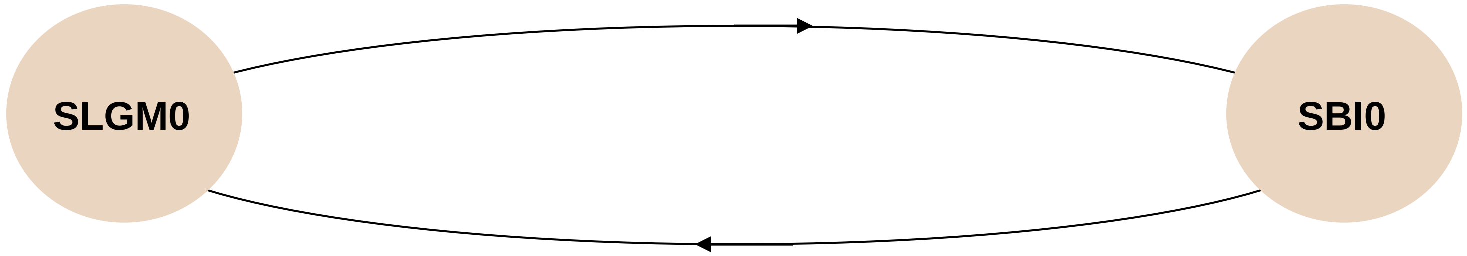
- Once the **initiator** has received the acknowledgment, the **FIX session is established** and trading can begin

Logon Message

- The Logon message **authenticates** a party establishing a connection to another system
- **It must be the first message** sent by the party requesting to initiate a FIX session
- **HeartBtInt (tag 108)** field is used to declare the timeout interval (how long we are idle outbound) before a Heartbeat is sent
- Session initiator can also use **ResetSeqNumFlag 141=Y** in Logon to reset sequence numbers to 1

Sample FIX Message – Logon and Logon Ack

8=FIX.4.2 9=0060 35=A 34=1 49=SLGM0 56=SBI0 52 =20070315-14:58:07 98=0 108=30 10=008



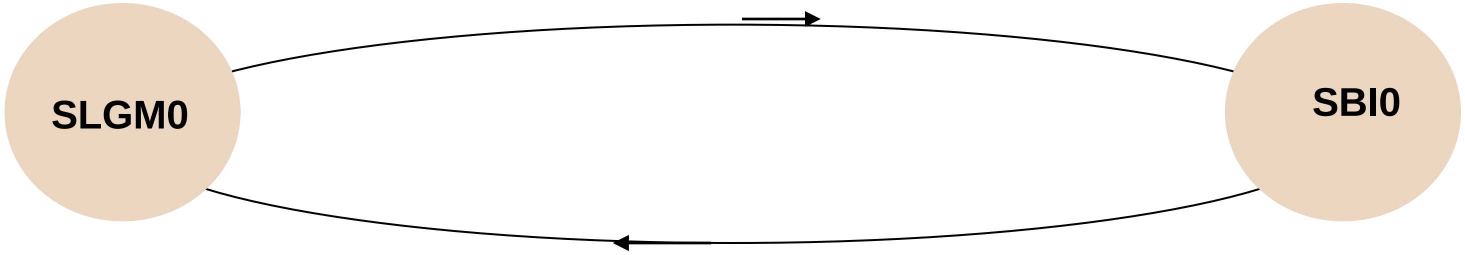
8=FIX.4.2 9=0060 35=A 34=1 49=SBI0 56=SLGM0 52 =20070315-14:58:08 98=0 108=30 10=009

FIX Support for 24 x 7 Trading

- As of FIX 4.1, FIX provides for a way of allowing 24-hour by 7-day trading (reset sequence numbers but stay connected)
- **ResetSeqNumFlag(141)=Y** in Logon message
- **Must also have MsgSeqNum(34)=1**
- Helps synchronize End-Of-Day (EOD) processing with counterparty

Session Reset using Logon with 141=Y

8=FIX.4.29=0066 35=A 34=1 49=SLGM0 56=SBI0 52=20070321-07:13:4598=0 108=30 141=Y10=051



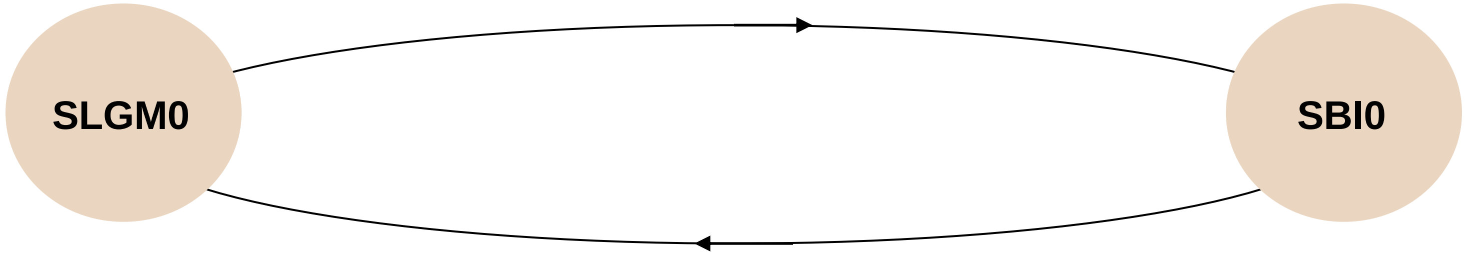
8=FIX.4.29=0066 35=A 34=1 49=SBI0 56=SLGM0 5 2=20070321-07:13:4598=0 108=30 141=Y10=051

What is a Heartbeat message?

- A **Heartbeat** message is a simple way to know that there is a connection to a counterparty is **still up and running**
- A Heartbeat is sent at a **pre-determined interval** (typically every 30 seconds) and is bi-directional (both sides send). This interval is set in the Logon message.
- If **no Heartbeat** is received within the heartbeat interval, the FIX engine should automatically **issue a Test Request**

Sample FIX Message – Heartbeat

8=FIX.4.2 9=0051 35=0 34=1928 49=SLGM0 56=SBI0 52=20070316-07:01:29 10=124



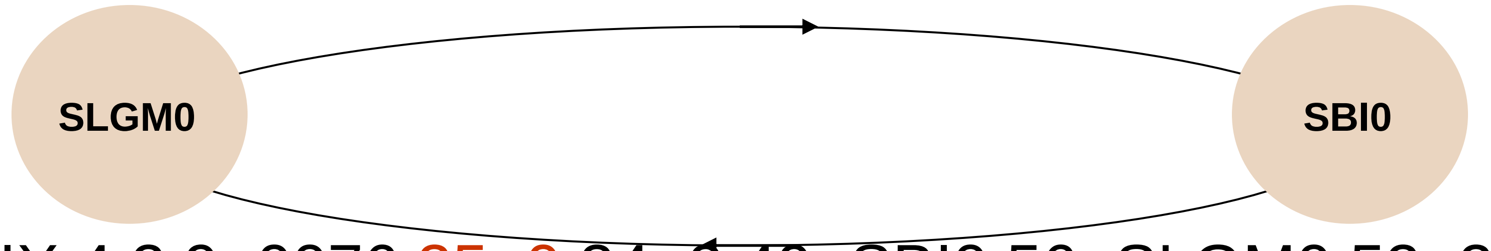
8=FIX.4.2 9=0051 35=0 34=1929 49=SBI0 56=SLGM0 52=20070316-07:01:32 10=119

What is a Test Request message?

- Either party on the FIX connection **can send a Test Request message at any time** during the FIX session
- Normally **sent automatically by the FIX engine** if a message has not been received within the heartbeat interval
- The recipient of a Test Request message **must respond with a Heartbeat** message
- The Test Request contains a required **TestReqID(112)** field. The Heartbeat response message **must contain** the TestReqID of the Test Request
- If the Test Request does not elicit a response – either continue sending Test Requests, or disconnect the counterparty

Sample FIX Messages – Test Request with Heartbeat Response

8=FIX.4.2 9=0076 35=1 34=2 49=SLGM0 56=SBI0 52=20070316-07:14:39 112=Appia-20070316-07:14:39 10=061



8=FIX.4.2 9=0076 35=0 34=2 49=SBI0 56=SLGM0 52=20070316-07:14:39 112=Appia-20070316-07:14:39 10=060

What is a sequence number?

- The most important concept to understand in the FIX Session layer
- It is a **unique identifier** for each message **in each direction** and is used by the FIX engine to track continuity of the session
- It is an **incrementing** whole number
- Normally at the **beginning of a trading day** the first message sent out by both sides would have sequence number **1 and** would increase for every new message

Message Sequencing

- It is the responsibility of both parties of a FIX connection to track sequence numbers
- Each party is **expecting a certain sequence number** from the other, which also **increments** for each message received
- Very important concept – **sequence numbers are unique for each connection** for each direction: inbound and outbound

Message Sequencing

- FIX assumes complete ordered delivery of messages between parties
- Implementers should consider this when designing message recovery and gap fill processes
- Two options exist for dealing with gaps:
  - Request all messages subsequent to the last message received
  - Request the specific message missed while maintaining an ordered list of all newer messages.

Possible Duplicates & Possible Resends

- FIX requires each party to mark any **possibly duplicate** messages and **track sequence numbers**

- FIX provides 2 methods to signal to a counterparty that this message **may** have been already sent under a previous sequence number:
  - **Possible Duplicate (PossDupFlag)** – used **by the FIX engine** to resend something under its **original sequence number** (because it has the message in its database)
  - **Possible Resend (PossResend)** – **your application uses this** because you are **not sure** whether your message made it to the engine. This uses **a new sequence number.**

Possible Duplicates & Possible Resends

- **PossResend(97)=**Y messages are **sent with NEW sequence numbers**

- In contrast, **PossDupFlag(43)**=Y messages are sent with the **original sequence number** – they are old messages

- PossResend messages are NEW messages as far as the FIX session is concerned – **the FIX engine doesn't know the difference** since the side that is resending the message believes **that the messages were already sent** once.
- (This is useful when an order remains unacknowledged for an inordinate length of time and the end-user suspects it had never been sent).

Tracking Message Sequence

- When the incoming sequence number does not match the expected number corrective processing is required:
  - If the incoming message has a sequence number **less than expected and the PossDupFlag(43) is not set, it indicates a serious error** and it is strongly recommended that the session be terminated
  - If the incoming sequence number is **greater than expected,** it indicates that messages were missed and **retransmission of the messages is requested**

Message Recovery

- FIX provides Session messages for
  - Requesting re-delivery of lost/missed messages through **Resend Request (35=2)**
  - Resynchronizing sequence numbers between parties through **Sequence Reset (35=4)**

- If a sequence number is missed, the party that did not receive the message must issue a **Resend Request**

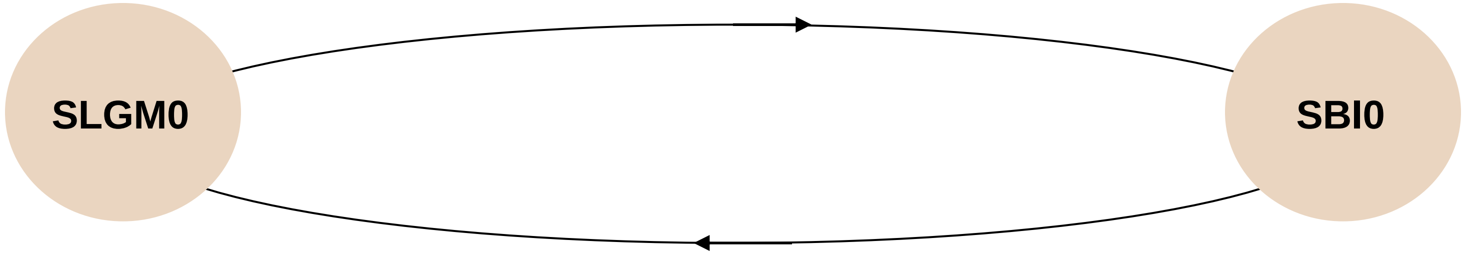
- To skip irrelevant messages, the re-sending party may issue a **Sequence Reset**

Resend Request Message

- Used to request the counterparty to resend a range of messages that you did not receive
  - **BeginSeqNo** (tag 7)
  - **EndSeqNo** (tag 16)
- Every message that is sent in response to a Resend Request must be labeled as a **“Possible Duplicate” (using PossDupFlag tag 43=Y)**
- The side **receiving** the resent messages must **realize that these are possible duplicates**, and process them accordingly
- **The first stage** of checking the possible duplicate is **by the FIX engine** – checks its database to see if it has the message
- Messages re-sent in response to Resend Request must also set tag **OrigSendingTime(122)** – the time of the original message

Sample FIX Messages – Resend Request and Possible Duplicate (PossDup) Replies

8=FIX.4.2 9=0057 35=2 34=5 49=SBI0 56=SLGM0 52=20070316-07:05:27 7=2 16=3 10=101



8=FIX.4.2 9=0159 35=D 57=Simulator 34=2 49=SLGM0 56=SBI0 43=Y 52=20070316-07:05:28 122=20070316-07:03:36 55=IBM 40=2 38=1000 21=2 11=order 0 60=20000124-10:10:10 54=1 44=110.5 10=157

What about sequence numbers and Logons?

There are three possible scenarios:

- #1: The incoming FIX Logon has the sequence number **the same** as I was expecting – all is good

- #2: The incoming FIX Logon has a sequence number **greater than** I was expecting – need to gap fill (Resend Request)

- #3: The incoming FIX Logon has a sequence number **less than** I was expecting – need to terminate the connection

And if the sequence number in the Logon was more than expected?

- If I was expecting Merrill Lynch to log in to me with a Logon of sequence number 100, but they come in with a Logon with a sequence number of 110

- I know that I have missed some messages, so then I would send a **Resend Request** for all the messages beginning with 100 (the first message that I didn’t get) onward

And if the sequence number in the Logon was less than expected?

- I was expecting Merrill Lynch to log in to me with a logon of sequence number 100, but they come in with a logon with a sequence number of 1 (most likely caused by them running EOD at the wrong time).

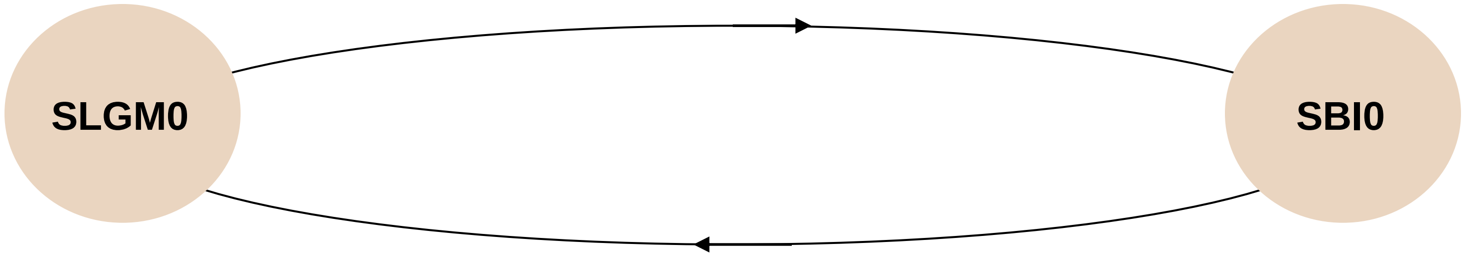
- This is a **serious communication error,** since Merrill Lynch has somehow decreased their sequence number without telling me.

- This **requires me to break the connection** and not accept any messages from them.

- I will need to call up ML and talk to the FIX engine administrator so that we can properly set the sequence numbers.

Sample FIX Messages – Logout in reply to Logon with MsgSeqNum less than expected

8=FIX.4.29=0060 35=A 34=1 49=SLGM0 56=SBI0 52=20070316-07:11:30 98=0 108=30
10=252



8=FIX.4.2 9=0146 35=5 34=7 49=SBI0 56=SLGM0 52=20070316-07:11:30 58=TEST_5403- Serious Error: Message sequence number: 1 is less than expected sequence number: 6 10=007

NYSE Technologies

How do Session-level Reject messages work?

- FIX provides a mechanism that allows a party to **reject an invalid message with Session Reject (35=3)**

- A message could be rejected for many possible reasons:
  - The message is **garbled or corrupted**
  - The message **does not pass the integrity check** based upon the CheckSum field in the trailer
  - **A required field in the message is missing** (for example, if I received an Order with no Symbol field set)
  - **A field in the message is improperly formatted** (for example, if I received an Order with an OrderQty field set with a non-numeric value)

How do Session-level Reject messages work?

- The FIX engine will create **a Session-level Reject message in response** to the invalid message and send it to the offending counterparty

- The **Text field (tag 58)** normally **identifies the reason for rejection**

Sequence Reset (Gap Fill mode)

- During the gap fill process, administrative messages should not be retransmitted
- **The SeqReset-GapFill (35=4)** may be used to **skip messages** that the sender chooses not to retransmit (e.g. administrative messages and aged orders)
- Logon, Logout, ResendRequest, Heartbeat, TestRequest and SeqReset-Reset and SeqReset-GapFill messages should not be resent
- Reject is the only administrative message which can be resent

Sequence Reset (Gap Fill) example

- If an Order is sent with sequence number 100, followed by two Heartbeats with sequence numbers 101, 102, and another Order with sequence number 103
- If the counterparty fails to receive all of these messages, it will send a Resend Request for 100 onward
- Upon receiving the Resend Request the engine should resend the Order with sequence number 100
- However, since 101 and 102 are Heartbeats, there is no real purpose to resend these
- The engine would instead send a Sequence Reset (Gap Fill), letting the counterparty know that the next message to be sent will be sequence number 103 (the Order the counterparty did not receive)

Sample FIX Messages – Sequence Reset (Gap Fill)

SequenceReset

8=FIX.4.29=0086 35=4 34=3 49=SLGM0 56=SBI0
43=Y 52=20070316-07:05:28 122=20070316-
07:05:28 36=4 123=Y 10=045

NewSeqNo

GapFillFlag

Ending a Session (Logout)

- Either side sends a ***Logout*** message (typically Initiator):

  8=FIX.4.2 9=0061 35=5 34=5 49=SLGM0 56=SBI0 52=20070 316-07:05:32 58=END_OF_DAY_ 11012008 10=247

- The other party responds with its own ***Logout***:

  8=FIX.4.2 9=0078 35=5 34=6 49=SBI0 56=SLGM0 52=20070 316-07:05:32 58=END_OF_DAY_ 11012008- Logout accepted 10=032

Review

- Delivery order of FIX messages is indicated by which tag?

  MsgSeqNum (34)

- Which type of message is used to indicate that the connection is up, even though no activity is happening over it?

  Heartbeat (35=0)

- If no message is received within the heartbeat interval, what type of message should be sent?

  Test Request (35=1)

- Does a Logon message have to be acknowledged?

  Yes

- Must a Heartbeat message be acknowledged?

  No

Review

- Must a Test Request message be acknowledged?

  Yes

- If a sequence number is missed, what should the automatic response be?

  Send a Resend Request message (35=2)

- Which message is used to disconnect a FIX session?

  Logout (35=5)

- If you receive a sequence number less than what you were expecting, what should the response be?

  Terminate the connection and contact the counterparty

- Which tag and value must be set in a Logon message to request the sequence numbers be reset?

  141=Y

Review

- Messages that are resent in response to a Resend Request must be labeled as such using which tag and value?

  PossDupFlag (43=Y)

- When your application is sending a message containing information that may have been sent before, how must the message be labeled?

  PossResend (97=Y)

- True or False: For a message with PossResend(97)=Y, its sequence number will be NOT the same as when the message was originally sent

  True

- Are Heartbeats and Test Requests resent in response to a Resend Request?

  No

- Name three session message types that must be acknowledged (most likely by the FIX engine itself)

  Logon, Resend Request, Logout

Review

Let’s consider this scenario:

Connection: SBI0

Status: DOWN

Message Sequence # (in¥out): 8¥8

Message exchange:

To SBI0: 8=FIX.4.2 9=0060 35=A 34=8 49=SLGM0 56=SBI0

52=20070906-15:31:34 98=0 108=30 10=013

From SBI0: 8=FIX.4.2 9=0060 35=A 34=1 49=SBI0
56=SLGM0 52=20070906-15:31:34 98=0 108=30 10=006

What happens next?

SLGM0 sends Logout (35=5) and breaks the connection to SBI0 since its next expected in sequence number is 8

Review

What, if anything, is wrong with these messages?

8=FIX.4.2 9=0076 35=1 34=2 49=SLGM0 56=SBI0 52=20070906-15:27:57 10=077

Required field TestReqID(112) is missing

8=FIX.4.2 9=0066 35=A 34=7 49=SLGM0 56=SBI0 52=20070906-15:35:15 98=0 108=30 141=Y 10=060

When sending ResetSeqNumFlag (141=Y), MsgSeqNum should be 1 (34=1)

8=FIX.4.2 9=0058 35=2 34=6 49=SLGM0 56=SBI0 52=20070906-15:37:14 7=12 16=10 10=162

BeginSeqNo (7=12) is greater than EndSeqNo (16=10)

8=FIX.4.2 9=0087 35=4 34=5 49=SBI0 56=SLGM0 43=Y 52=20070906-15:37:15 123=Y 36=15 10=108

There is nothing wrong

Review

Scenario:

Connection: SBI0
Status: DOWN
Message Sequence # (in¥out): 5¥5

Message exchange:

To SBI0: 8=FIX.4.2 9=0060 35=A 34=5 49=SLGM0 56=SBI0 52=20070906-15:37:14 98=0 108=30 10=014

From SBI0: 8=FIX.4.2 9=0061 35=A 34=15 49=SBI0 56=SLGM0 52=20070906-15:37:14 98=0 108=30 10=064

What happens next?

SLGM0 sends a Resend Request with BeginSeqNo (7=5) and EndSeqNo (16=14)

NYSE Technologies SM

Powering the exchanging world. SM

Module VI: The FIX Application Layer

The FIX Application Layer

Topics in this module:

- Functionality of the FIX Application layer
- Orders and Executions – FIX message format and required fields
- Order flow (Executions, Cancels)
- Considerations when sending FIX messages
- User-defined fields and messages
- Repeating fields and groups

What is the FIX Application Layer?

An important distinction to be aware of is the difference between the FIX Session layer and the Application layer:

- **Session layer** – handles communication between the two parties and ensures that no messages are missed and bad messages are rejected
    - Logon, Logout, Heartbeat, Test Request, Resend Request, etc.

- **Application layer** – contains the actual useful financial data that the business will care about
    - Order, Execution Report, Indication of Interest, Allocation, Quote, Order Cancel Request, etc.

What does the Application Layer do?

- FIX provides format and rules for transmitting financial information
  - Defines the messages, their names, application to the business, and for each message which tags are required, optional, and conditionally required in that message's Body
  - Defines the fields, their names, meaning, format, and bounds

- For example, a FIX Order message must contain certain mandatory fields such as Symbol, Side, Quantity, and Order Type

What does the Application Layer do?

- The actual checking of the validity of a FIX message is done by the FIX engine itself

- Any FIX engine that you purchase or develop yourself must be able to follow the rules of FIX when checking inbound and outbound messages

- FIX provides the rules for the message structure, but it is up to the FIX engine to check these rules

What does the Application Layer Not Provide?

- FIX does not provide rules for rejecting messages that are valid FIX but incorrect in some other capacity

- For example, if I receive a Cancel Request for an Order I never received, FIX does not automatically say that message should be rejected by the FIX engine

- Rejection of a message of this kind must be done by a client program or a trading system that is interfacing with the FIX engine

Types of Application Messages

Pre-trade

- Indication, Event Communication, Quotation,
- Market Data, Security & Trading Session Definition/Status

Orders & Executions (Trade)

- Single/General Order Handling, Cross Orders,
- Multi-leg Orders (Swaps, Option Strategies, etc),
- List/Program/Basket Trading

Post-trade

- Allocation and Ready-To-Book, Settlement Instructions,
- Trade Capture (“Streetside”) Reporting, Registration Instructions

Application Messages – from Buy-side

- Quote Request
- Order, Cancel, Modification (Cancel/Replace) Requests
- Allocation Instructions
- Email
- List Order / Program / Basket Trading
- Market Data Request, Security Definition Request, Security Status Request, Trading Session Status Request, etc.

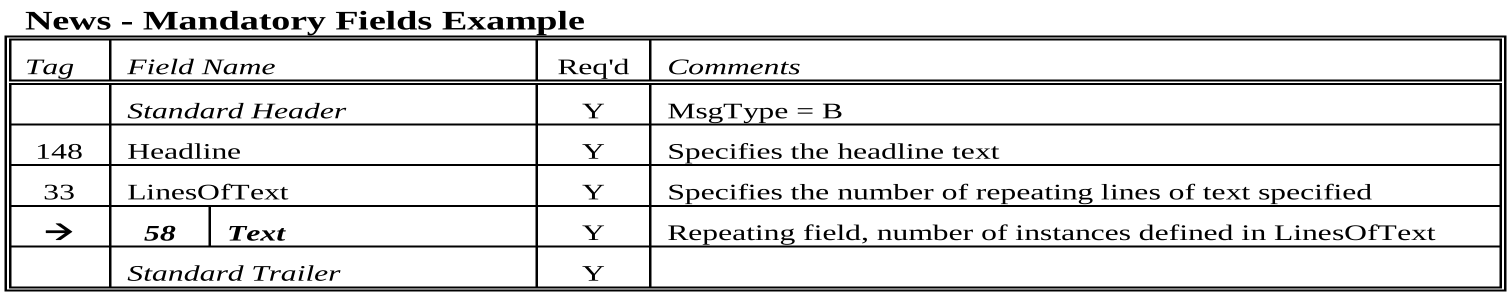
Application Messages – from Sell-side

- Indication of Interest, News, Email
- Post-Trade Advertisement
- Quotes
- Order Acknowledgment, Change Acknowledgment (Cancels, Modifications)
- Executions – Partial Fill, Complete Fill, Done For Day
- Market Data, Security Definition, Security Status, Trading Session Status, Mass Quote, etc

Repeating Fields

- FIX allows for certain fields to occur more than once in a message

- Simplest example – FIX News message (35=B)



News - Mandatory Fields Example

Tag	*Field Name*	Req'd	*Comments*
	Standard Header	Y	MsgType = B
148	Headline	Y	Specifies the headline text
33	LinesOfText	Y	Specifies the number of repeating lines of text specified
→	***58*** ***Text***	Y	Repeating field, number of instances defined in LinesOfText
	Standard Trailer	Y	

- LinesOfText (tag 33) indicates how many times Text (tag 58) will appear in the message
- 8=FIX.4.29=013835=B34=23449=ABC56=XYZ52=20070318-11:16:56148=HDLINE33=258=Line158=Line210=252

Repeating Groups

- A group of fields can also repeat
- Using New Order – List (35=E), for example, a basket of orders can be expressed with a single FIX message
- NoOrders (tag 73) – called a repeating group indicator – is a special field that tells us how many groups we should expect to follow
- Fields ClOrdID, Symbol, and Side will then repeat for each order in the list

Repeating Group Rules

- A repeating group is always preceded by its “indicator”, No<Group>, which tells us how many group instances will follow
- If a group is present at all, then the first field of that group (“leading tag”) is always required
- If a group field is listed as required, then it must appear once in every group instance
- Repeating group fields must appear, if appearing at all, in the order listed in the FIX specification
- Repeating groups may be nested within one another – for example, in the New Order – List message, a NoAllocs(78) group could appear within one or more instances of the NoOrders(73) group

List Order Example

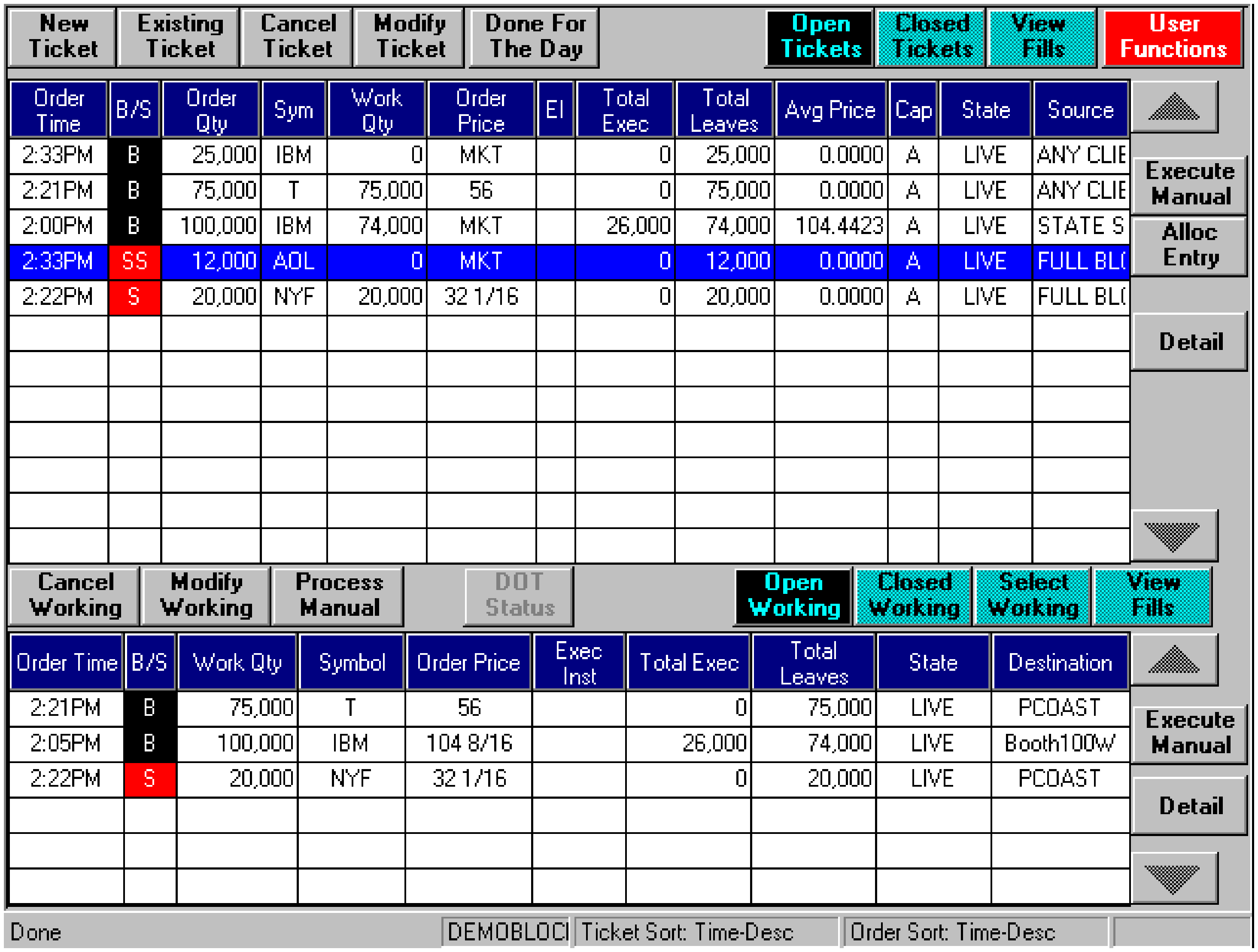
```
8=FIX.4.29=15753 5=E128=RECVGBRKR34=32349=INVMGR56=HUB52=20070416-
07:37:0950=EDWI66=36436966394=368=1073=1011=3643696667=121=3100=S55=CH001057076748=CH00
1057076722=4207=S106=Chocoladefabriken Lindt & Spr<FC>107=Chocoladefabriken Lindt &
Spru54=238=10040=115=CHF59=011=3643696767=221=3100=VX55=CH001103746948=CH001103746922
=4207=VX106=Syngenta AG107=Syngenta
Reg.54=238=300040=115=CHF59=011=3643696867=321=3100=VX55=CH001205604748=CH001205604722
=4207=VX106=Nestle S.A.107=Nestl<E9>
Reg.54=238=100040=115=CHF59=011=3643696967=421=3100=VX55=CH001225515148=CH001225515122
=4207=VX106=The Swatch Group AG107=The Swatch
Group54=238=100040=115=CHF59=011=3643697067=521=3100=D55=DE000555200448=DE000555200422
=4207=D106=Deutsche Post AG107=Deutsche Post
Reg.54=238=1000040=115=EUR59=011=3643697167=621=3100=F55=DE000515100548=DE000515100522
=4207=F106=BASF
AG107=BASF54=238=500040=115=EUR59=011=3643697267=721=3100=F55=DE000578580248=DE00057
8580222=4207=F106=Fresenius Medical Care AG & Co107=Fresenius Medical
Care54=238=500040=115=EUR59=011=3643697367=821=3100=T55=JP390290000448=JP390290000422=4
207=T106=Mitsubishi UFJ Financial Group107=Mitsubishi UFJ Financial
Group54=238=5040=115=JPY59=011=3643697467=921=3100=O55=US594918104548=US594918104522=4
207=O106=Microsoft Corp.107=Microsoft
Corp.54=238=1000040=115=USD59=011=3643697567=1021=3100=VX55=CH002489948348=CH002489948
322=4207=VX106=UBS AG, Zurich107=UBS Reg.54=238=1000040=115=CHF59=010=187
```

Are there any messages that must be acknowledged?

Several messages in the Application layer must be acknowledged:

- New Order
- Order Cancel Request
- Order Cancel/Replace Request
- Order Status Request
- Allocation

Sample Order Blotter



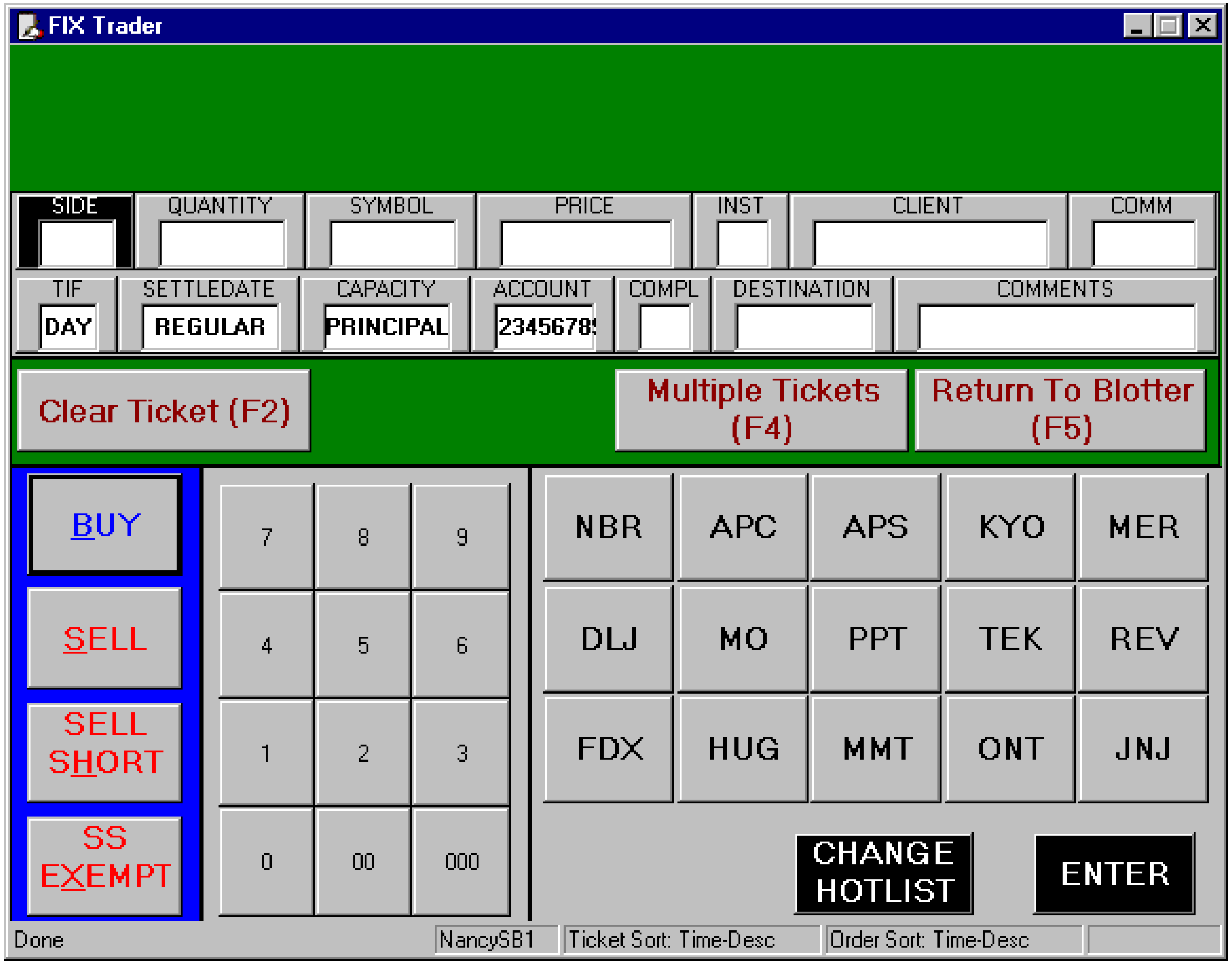
Order Time	B/S	Order Qty	Sym	Work Qty	Order Price	EI	Total Exec	Total Leaves	Avg Price	Cap	State	Source
2:33PM	B	25,000	IBM	0	MKT		0	25,000	0.0000	A	LIVE	ANY CLIE
2:21PM	B	75,000	T	75,000	56		0	75,000	0.0000	A	LIVE	ANY CLIE
2:00PM	B	100,000	IBM	74,000	MKT		26,000	74,000	104.4423	A	LIVE	STATE S
2:33PM	SS	12,000	AOL	0	MKT		0	12,000	0.0000	A	LIVE	FULL BL(
2:22PM	S	20,000	NYF	20,000	32 1/16		0	20,000	0.0000	A	LIVE	FULL BL(

Order Time	B/S	Work Qty	Symbol	Order Price	Exec Inst	Total Exec	Total Leaves	State	Destination
2:21PM	B	75,000	T	56		0	75,000	LIVE	PCOAST
2:05PM	B	100,000	IBM	104 8/16		26,000	74,000	LIVE	Booth100W
2:22PM	S	20,000	NYF	32 1/16		0	20,000	LIVE	PCOAST

Done | DEMOBLOC| | Ticket Sort: Time-Desc | Order Sort: Time-Desc

NYSE Technologies

Sample Order Entry Grid

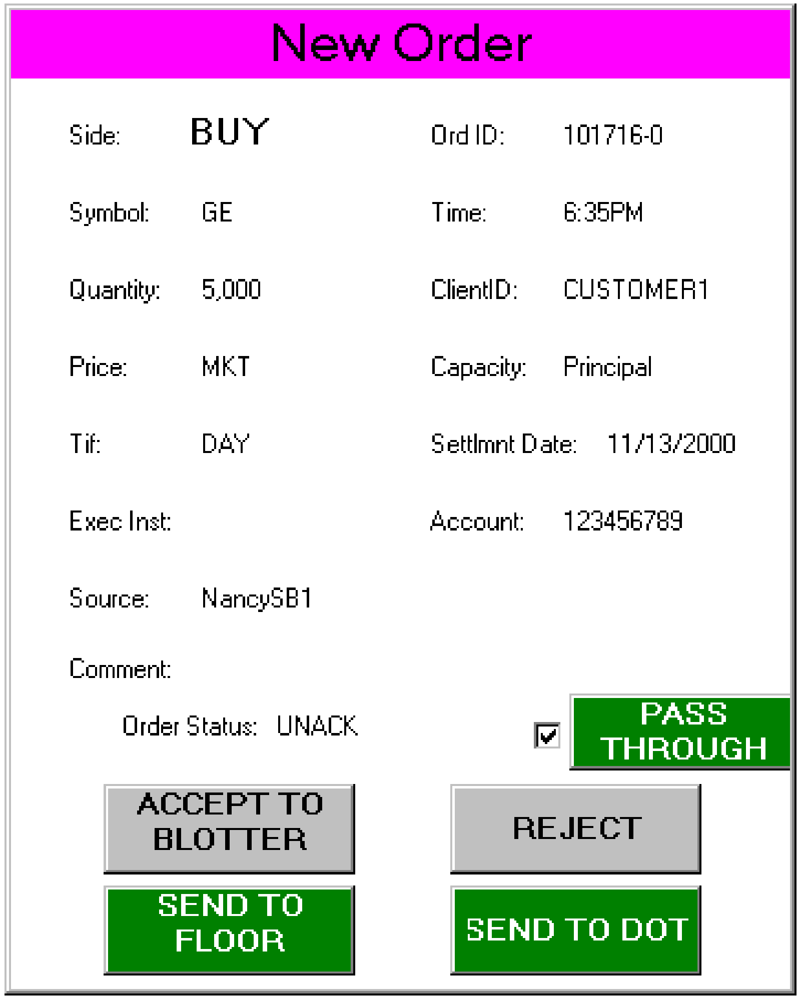

FIX Trader

SIDE	QUANTITY	SYMBOL	PRICE	INST	CLIENT	COMM

TIF	SETTLEDATE	CAPACITY	ACCOUNT	COMPL	DESTINATION	COMMENTS
DAY	REGULAR	PRINCIPAL	23456789			

Done | NancySB1 | Ticket Sort: Time-Desc | Order Sort: Time-Desc

Sample Order Pop-up

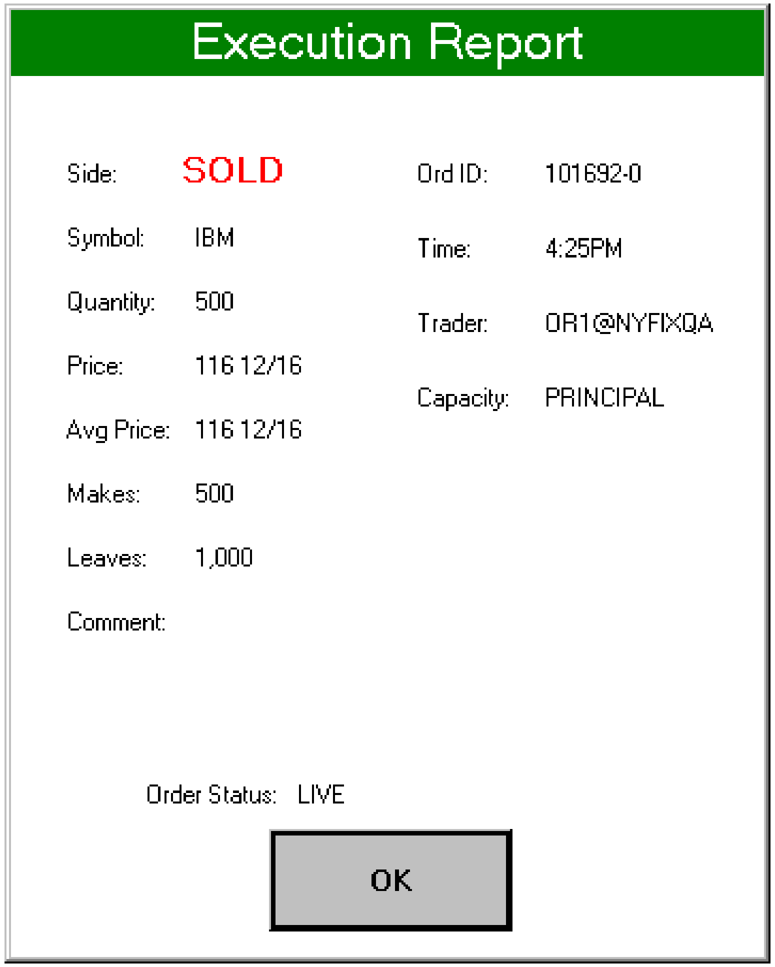


New Order

Side:	BUY	Ord ID:	101716-0
Symbol:	GE	Time:	6:35PM
Quantity:	5,000	ClientID:	CUSTOMER1
Price:	MKT	Capacity:	Principal
Tif:	DAY	Settlmnt Date:	11/13/2000
Exec Inst:		Account:	123456789
Source:	NancySB1		
Comment:			

Order Status: UNACK

Sample Execution / Order Fill


Execution Report

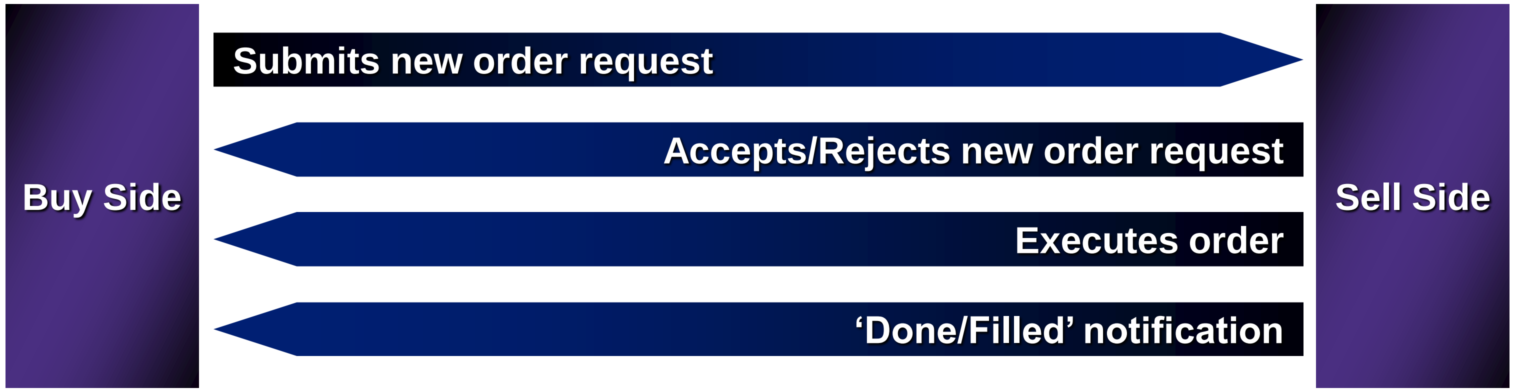
Side:	SOLD	Ord ID:	101692-0
Symbol:	IBM	Time:	4:25PM
Quantity:	500	Trader:	OR1@NYFIXQA
Price:	116 12/16	Capacity:	PRINCIPAL
Avg Price:	116 12/16		
Makes:	500		
Leaves:	1,000		
Comment:			

Order Status: LIVE

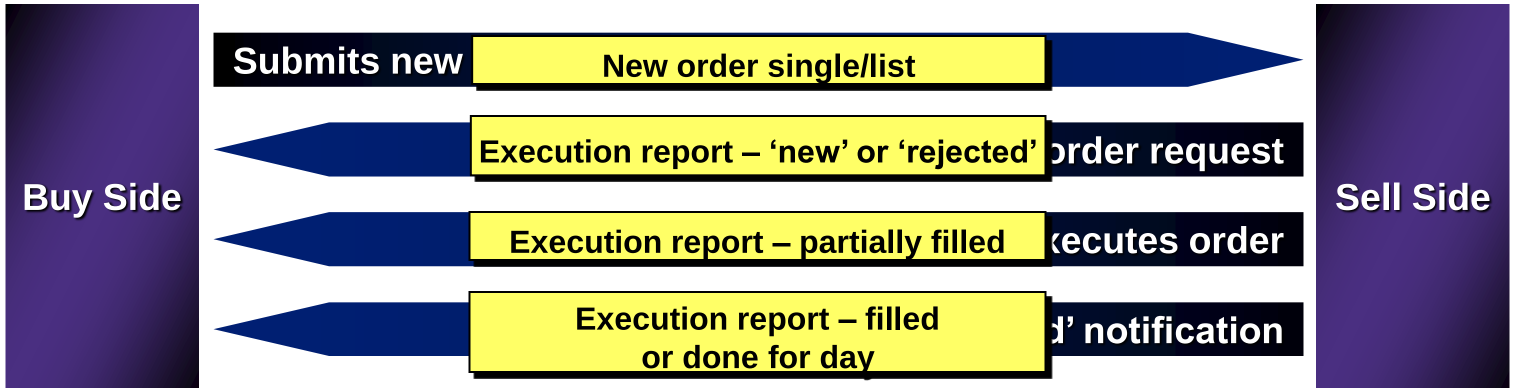
How do I create an Order in FIX?

- The following sections detail the creation of a FIX Order (**New Order – Single**) message
- **All the examples** that will be given in the following section are in **FIX 4.2**, as it is the most common version of the FIX protocol in use today (however, it is not the most recent version of the FIX protocol)

Conceptual Order Processing Flow New Order


Buy Side
Submits new order request
Accepts/Rejects new order request
Executes order
'Done/Filled' notification
Sell Side

With FIX Messages


Buy Side
Sell Side
Submits new
New order single/list
Execution report – ‘new’ or ‘rejected’
order request
Execution report – partially filled
xecutes order
Execution report – filled or done for day
d’ notification

NYSE Technologies

What fields does an order message require?

- In FIX, each message has required, optional and conditionally required fields.

- Required fields for a message **must be set.**

- In FIX 4.2, Order message must have following tags:
    - Client Order ID (11); Handling Instruction (21); Symbol (55); Side (54); Transaction Time (60); Order Type (40)

What is the Client Order ID used for?

- ClOrdID (tag 11) is a unique identifier for any order
- This field helps to keep track of multiple orders over the course of the day
- ClOrdID must be unique throughout at least a particular trading day
- This field is assigned by the Order Management System (OMS). A simple incrementing value is probably easiest

What are Handling Instructions in an Order?

- **HandlInst (tag 21)** specifies whether the order should be automatically executed or if broker intervention is required

- There are three possible values for this
  - 1 (automatic)
  - 2 (automatic with possible broker intervention)
  - 3 (completely manual execution)

- Most commonly this will be ‘1’ – automatic trade execution with no broker intervention

What are Symbol, Side, & Transaction Time?

- Symbol (tag 55) – the stock symbol for that security (European stocks additionally may use tags 48 (SecurityID) and 22 (IDSource) to identify the instrument relative to ISIN or SEDOL, etc.)

- Side (tag 54) - represents whether a stock is being bought or sold – a value of ’1’ indicates that it is a BUY order, and a value of ‘2’ indicates it is a SELL order

- TransactTime (tag 60) is the time the order was created by the trader or the trading system

Purpose of Order Quantity and Order Type

- OrderQty (tag 38) – the number of shares being bought or sold in that order

- OrdType (tag 40) – Specifies type of order – for example, if we want to buy Intel at the market price, then OrdType is set to ‘1’ to indicate a market order

- Other values for OrdType could be ‘2’ for a limit order, ‘3’ for a stop order, ‘4’ for a stop limit order, etc.

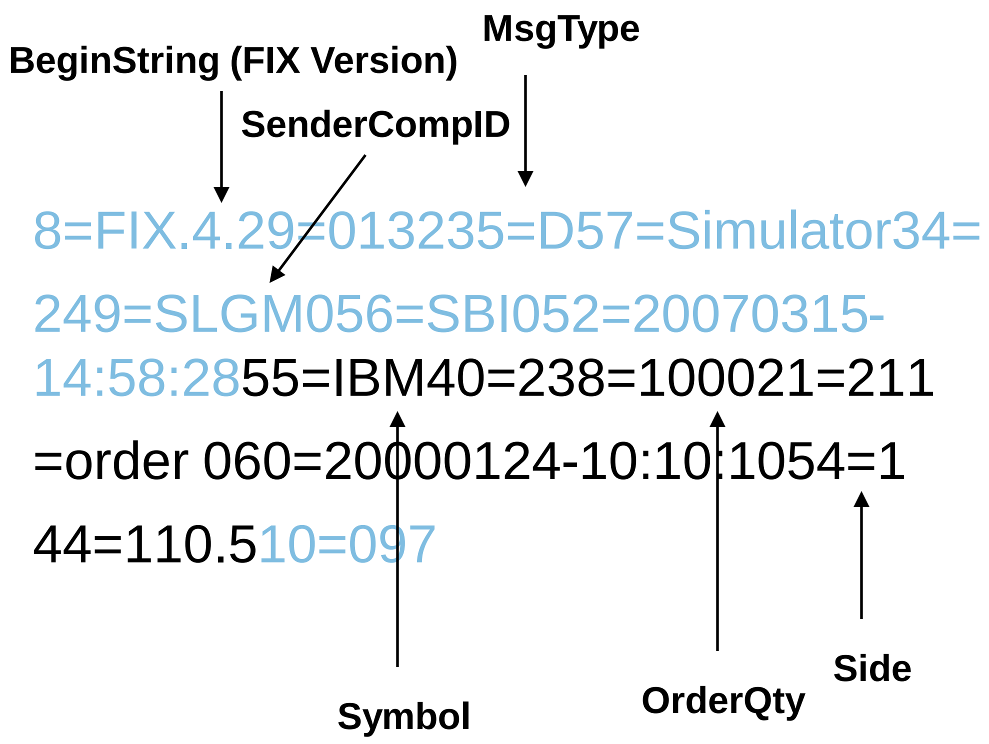
Are there important optional fields?

- Price (tag 44) – Though not required for market orders, the Price field is required when sending a limit order

- TimeInForce (tag 59) – specifies the duration for which this order is valid. By default, the value of this field is “Day”. Other possible values include ‘1’ Good Till Cancel (GTC), ‘2’ At the Opening, and ‘4’ Fill or Kill, etc.

- Text (tag 58) – Each FIX message has a Text field which allows a user to put in any free form text

The New Order Message



```
8=FIX.4.29=013235=D57=Simulator34=
249=SLGM056=SBI052=20070315-
14:58:2855=IBM40=238=100021=211
=order 060=20000124-10:10:1054=1
44=110.510=097
```

How is an Order message acknowledged?

- The sell-side (the side that is receiving this order) must immediately acknowledge receipt of the order by sending an Execution Report (35=8) message

- Subsequent fills and rejects should be sent in separate Execution Report messages after order acknowledgment

The Execution Report Message

The execution report message is used to:

- Confirm the receipt of an order
- Confirm changes to an existing order (i.e. accept cancel and replace requests)
- Relay order status information
- Relay fill information on working orders
- Reject orders
- Report post-trade fees calculations associated with a trade

Required Fields in the Execution Report

Tag	*Field Name*	*Req'd*	*Comments*
	Standard Header	Y	MsgType = 8
37	OrderID	Y	OrderID is required to be unique for each chain of orders.
17	ExecID	Y	Must be unique for each Execution Report message
20	ExecTransType	Y	
150	ExecType	Y	Describes the type of execution report. Same possible values as OrdStatus.
39	OrdStatus	Y	Describes the current state of a CHAIN of orders, same scope as OrderQty, CumQty, LeavesQty, and AvgPx
55	Symbol	Y	
54	Side	Y	
151	LeavesQty	Y	Amount of shares open for further execution. If the OrdStatus is Canceled, DoneForTheDay, Expired, Calculated, or Rejected (in which case the order is no longer active) then LeavesQty could be 0, otherwise LeavesQty = OrderQty - CumQty.
14	CumQty	Y	Currently executed shares for chain of orders.
6	AvgPx	Y	
	Standard Trailer	Y	

Important tags in Execution Report message

- **OrderID (tag 37)** – Unique identifier for this order as assigned by the sell-side – this is how the sell-side keeps track of the orders that it is executing
- **ExecID (tag 17)** – Unique ID for this particular execution report. Note the difference with OrderID – one Order (with OrderID "Order1") might correspond with multiple Execution Reports, each with a different ExecID, but all will share the same OrderID
- **ExecType (tag 150)** – describes the type of execution – '0' for New, '1' for Partially Filled, '2' for Completely Filled
- **ExecTransType (tag 20)** – Execution Transaction Type – indicates the type of transaction – values include '0' for New, '1' for Cancel, '2' for Correct, '3' for Status – normally this will be New

Important tags in Execution Report message

- **OrdStatus (tag 39)** – tells about the overall status of the order at the time of report. This is in contrast to ExecType(150), which describes only the status of that specific execution. Here are a few possible values and their meanings:

0	New	4	Canceled
1	Partially filled	5	Replaced
2	Filled	8	Rejected
3	Done for day	C	Expired

- **Symbol, Side, and OrderQty** are used in the same manner as in the Order Message

- In the Execution Reports sent back for the Order, these fields must match the values that were in the original Order message

Important tags in Execution Report message

- **LastShares (tag 32**) – the number of shares that were filled by this Execution Report (for an ER that is an order acknowledgement, this will always be 0)

- **LastPx (tag 31)** – the price at which the shares in this particular execution report were filled (for an ER which is an acknowledgement, this will always be 0)

- **CumQty (tag 14)** – total number of shares filled for this order up to this point

- **AvgPx (tag 6)** – the average price per share for the portion of the order quantity that has been filled up to this point

- **LeavesQty (tag 151)** – the number of shares left unfilled

Quantity Calculations

- If the OrdStatus(39) is Canceled, Filled, DoneForDay, Expired, Calculated, or Rejected → THEN LeavesQty should be 0
- LeavesQty = OrderQty – CumQty
- If LeavesQty is not equal to 0, it means the order is still active and the buy-side should expect more execution reports

What other fields in the Execution Report should I be aware of?

- **ClOrdID** – the same field as in the Order message, and if set would have to match the corresponding Order's ClOrdID

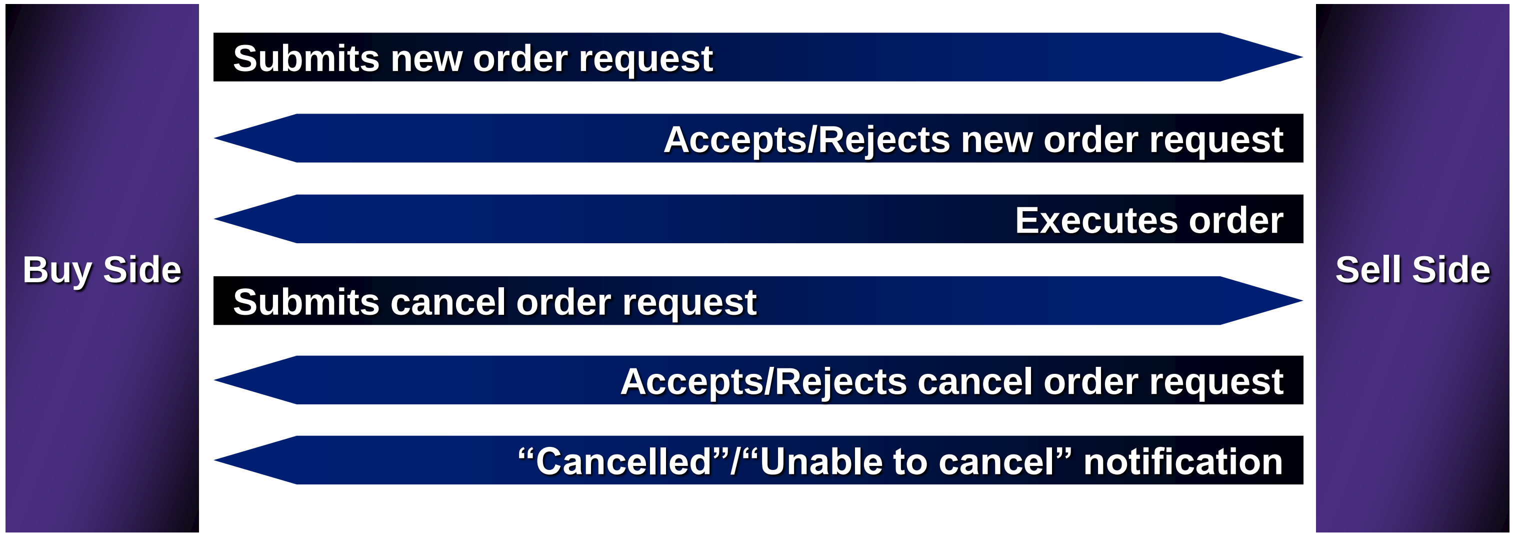
- The distinction between **OrderID(37) and ClOrdID(11)** is that:
  - ClOrdID is set by the buy-side (the side that originated the order) and is the buy-side's unique identifier for the order; whereas
  - OrderID is set by the sell-side (the side that received the order) – so the two sides each have their unique own identifier for the same order

- This field is not required ONLY for manually-entered orders (for electronic orders this must be set)

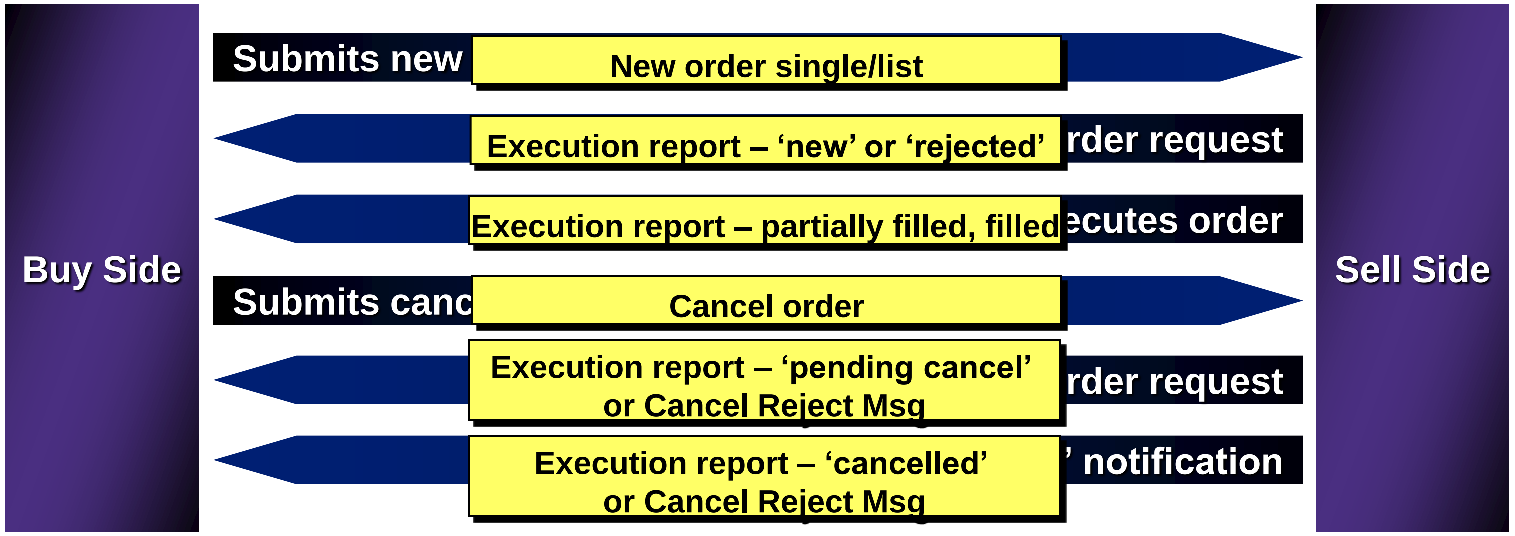
What if I want to cancel or change an Order?

- FIX provides a mechanism to cancel an order that has already been sent
- The order can be canceled completely by sending an Order Cancel Request
- Alternatively, some aspects of it can be modified (i.e. Quantity, Price) without canceling it outright by using the Order Cancel/Replace message

Conceptual Order Processing Flow Cancel Order


Buy Side
Submits new order request
Accepts/Rejects new order request
Executes order
Submits cancel order request
Accepts/Rejects cancel order request
“Cancelled”/“Unable to cancel” notification
Sell Side

With FIX Messages


Buy Side
Sell Side
Submits new
New order single/list
Execution report – 'new' or 'rejected'
rder request
Execution report – partially filled, filled
ecutes order
Submits canc
Cancel order
Execution report – 'pending cancel' or Cancel Reject Msg
rder request
Execution report – 'cancelled' or Cancel Reject Msg
' notification

Order Cancel Request (35=F)

- The Order Cancel Request message is used to cancel any **unfilled quantity** of an order
- This message must be used to cancel an order sent with an incorrect side or symbol
- **A new ClOrdID(11) should be assigned** to this message with a reference to ClOrdID of the original Order message in **OrigClOrdID(41)**
- The required fields in the Order Cancel Request are
  - OrigClOrdID, ClOrdID, Symbol, Side, OrderQty, TransactTime

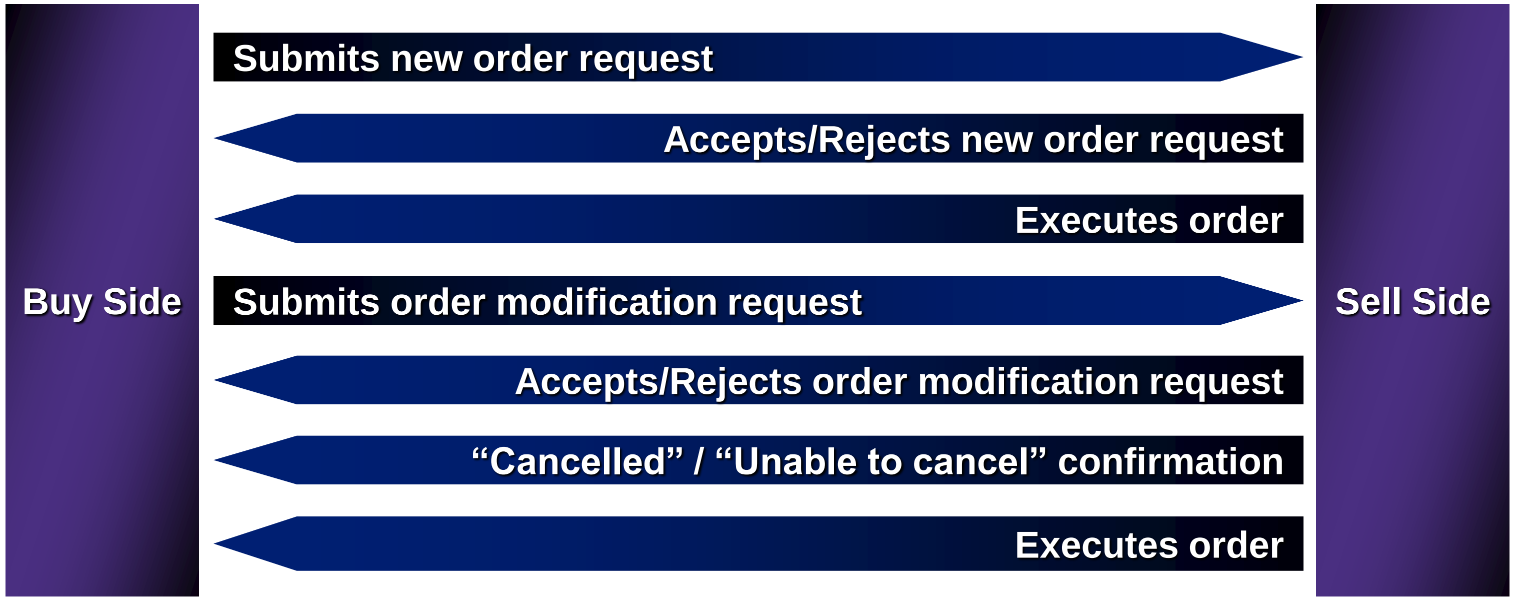
What are OrigClOrdID and ClOrdID?

- **OrigClOrdID (tag 41)** – Original Client Order ID – the Client Order ID of the original Order – you must set it or else the sell-side will not know which order you are canceling

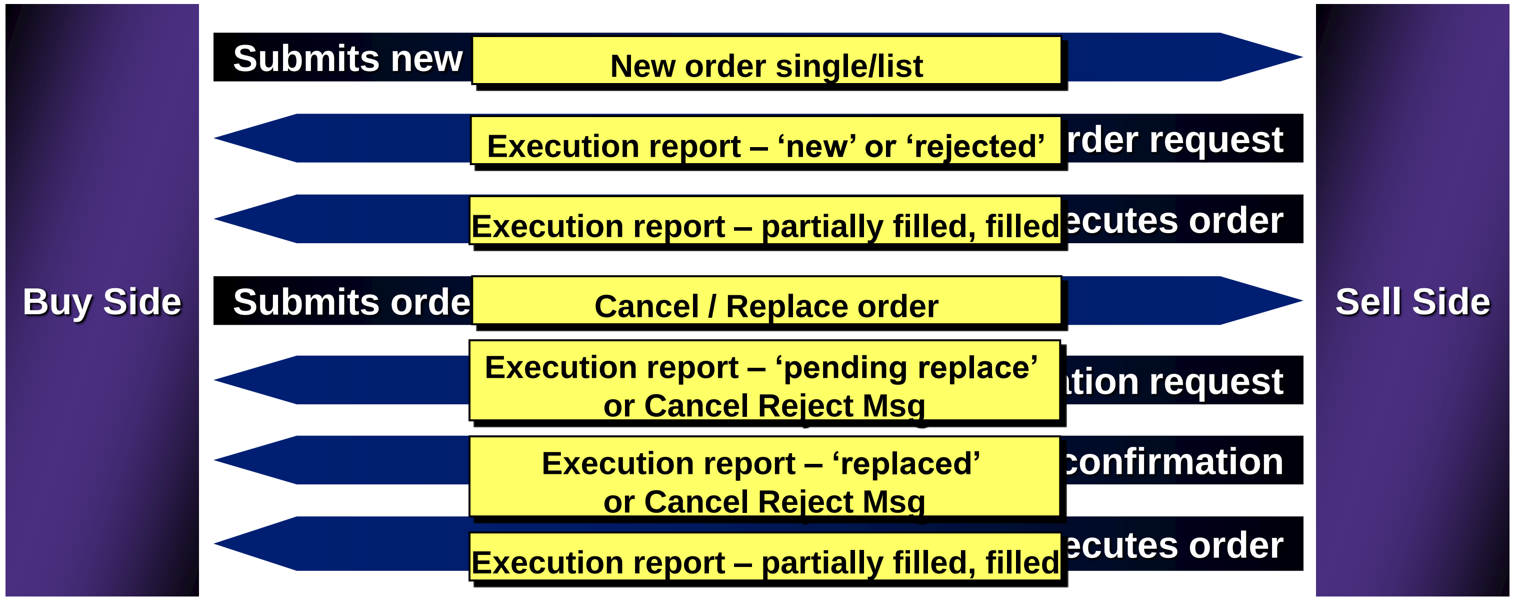
- **ClOrdID (tag 11)** – a unique identifier for this Order Cancel Request – must be a new identifier and not an identifier already used for a previous order

- Every Order Cancel Request and Order Cancel/Replace subsequent to the original Order should have a new ClOrdID value with reference to the previous in OrigClOrdID. This is called Order Chaining in FIX

Conceptual Order Processing Flow Modify Order


Buy Side
Submits new order request
Accepts/Rejects new order request
Executes order
Submits order modification request
Accepts/Rejects order modification request
“Cancelled” / “Unable to cancel” confirmation
Executes order
Sell Side

With FIX Messages


Buy Side
Sell Side
Submits new
New order single/list
Execution report – 'new' or 'rejected'
rder request
Execution report – partially filled, filled
ecutes order
Submits orde
Cancel / Replace order
Execution report – 'pending replace' or Cancel Reject Msg
ition request
Execution report – 'replaced' or Cancel Reject Msg
confirmation
Execution report – partially filled, filled
ecutes order

Order Cancel/Replace Request (35=G)

- The Order Cancel/Replace message is used to modify certain parts of an Order
- Some things which can be changed in an already-placed Order are
  - OrderQty, OrdType, Price (for Limit orders), TimeInForce, etc.
- These fields should be set to the NEW desired values

Order Cancel/Replace Request (35=G)

- For example, if we wanted to change the price of a limit order (to $54 instead of $55), we would send an Order Cancel/Replace message with Price field set to 54 (44=54)

- Fields like Side and Symbol cannot be changed. If Side or Symbol need to be changed, the existing Order should be canceled and new one placed

Other considerations in FIX trading

- Business semantics of FIX tags (security master and default currency to be used, for example)

- Your application / OMS should perform some additional validation on FIX messages' values

- If you are sending Orders
  - ClOrdID is very important – you have to have a robust system of keeping track of them over the course of the day

Other considerations in FIX trading

If you are sending Orders:

- Your application has to keep track of all the relevant data for the orders as well. For example, if you send an order with ClOrdID of 1, for 5000 shares of Yahoo, and fills start coming back for ClOrdID 1 for Amazon, you know that those fills are wrong.

Other considerations in FIX trading

- There is also the consideration of what to do if you send an Order and do not get an Execution Report acknowledgement back

- In that scenario, the application would not allow further work with that order since it is unacknowledged (so you would not allow a cancel, etc., to be sent for that order)

Sample Order Flow In FIX – New Order Single Buy-side Perspective

8=FIX.4.2 9=0136 35=D 57=Simulator 34=2 49=SLGM0 56=SBI0 52=20070318-09:33:07 55=IBM 40=2 38=2000 21=2 11=OrderNumber0 60=20070318-10:10:10 54=1 44=110.5 10=038

8=FIX.4.2 9=0212 35=8 50=Simulator 34=2 49=SBI0 56=SLGM0 52=20070318-09:33:08 151=2000 55=IBM 40=2 11=OrderNumber0 31=0.0 150=0 39=0 54=1 44=110.5 32=0 17=63187159 38=2000 21=2 60=20070318-09:33:08.73 76=0.0 20=0 14=0 37=63089743 10=173

8=FIX.4.2 9=0222 35=8 50=Simulator 34=3 49=SBI0 56=SLGM0 52=20070318-09:33:11 151=1000 55=IBM 40=2 11=OrderNumber0 31=110.5 150=1 39=1 54=1 44=110.5 32=1000 17=63187160 38=2000 21=2 60=20070318-09:33:11.73 76=110.5 20=0 14=1000 37=63089743 10=140

8=FIX.4.2 9=0224 35=8 50=Simulator 34=4 49=SBI0 56=SLGM0 52=20070318-09:33:14 151=0 55=IBM 40=2 11=OrderNumber0 31=110.375 150=2 39=2 54=1 44=110.5 32=1000 17=63187161 38=2000 21=2 60=20070318-09:33:14.73 86=110.4375 20=0 14=2000 37=63089743 10=017

Sent Order – A Closer Look

```
8=FIX.4.29=013635=D57=Simulator34=249=SLGM056=SBI052=20070318-
09:33:0755=IBM40=238=200021=211=OrderNumber0 60=20070318-
10:10:1054=144=110.510=038
```

35=D	**MsgType = D / Order**
49=SLGM0	**SenderCompID = SLGM0**
56=SBI0	**TargetCompID = SBI0**
55=IBM	**Symbol = IBM**
40=2	**OrdType = 2 / Limit**
44=110.5	**Price = 110.5**
38=2000	**OrderQty = 2000**
11=OrderNumber0	**ClOrdID = OrderNumber0**
54=1	**Side = 1 / Buy**

Received Execution Report – Acknowledgement
A Closer Look

35=8	MsgType = 8 / Execution Report
151=2000	LeavesQty = 2000
31=0.0	LastPx = 0.0
150=0	ExecType = 0 / New
39=0	OrdStatus = 0 / New
32=0	LastShares = 0
17=63187159	ExecID = 63187159
6=0.0	AvgPx = 0
20=0	ExecTransType = 0 / New
14=0	CumQty = 0
37=63089743	OrderID = 63089743

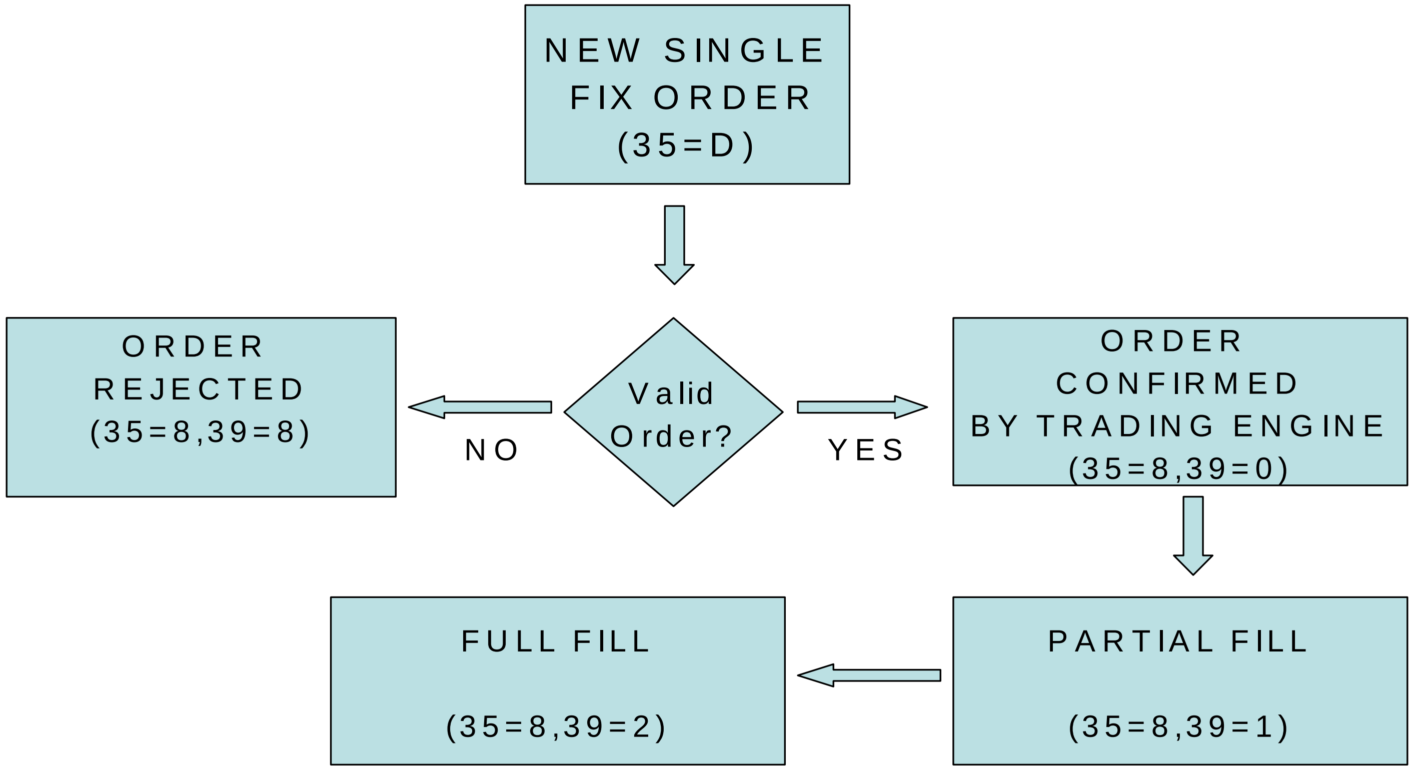
Received Execution Report – Partial Fill
A Closer Look

35=8	MsgType = 8 / Execution Report
151=1000	LeavesQty = 1000
31=110.5	LastPx = 110.5
150=1	ExecType = 1 / Partial Fill
39=1	OrdStatus = 1 / Partial Fill
32=1000	LastShares = 1000
17=63187160	ExecID = 63187160
20=0	ExecTransType = 0 / New
6=110.5	AvgPx = 110.5
14=1000	CumQty = 1000
37=63089743	OrderID = 63089743

Received Execution Report – Filled
A Closer Look

35=8	MsgType = 8 / Execution Report
151=0	LeavesQty = 0
31=110.375	LastPx = 110.375
150=2	ExecType = 2 / Fill
39=2	OrdStatus = 2 / Fill
32=1000	LastShares = 1000
17=63187161	ExecID = 63187161
6=110.4375	AvgPx = 110.4375
20=0	ExecTransType = 0 / New
14=2000	CumQty = 2000
37=63089743	OrderID = 63089743

Standard Order / Execution


NEW SINGLE FIX ORDER (35=D)
Valid Order?
NO
ORDER REJECTED (35=8,39=8)
YES
ORDER CONFIRMED BY TRADING ENGINE (35=8,39=0)
PARTIAL FILL (35=8,39=1)
FULL FILL (35=8,39=2)

Order Flow Matrix – Filled Order (D1)

Time	Message Received (ClOrdID, OrigClOrdID)	Message Sent (ClOrdID, OrigClOrdID)	Exec Type	OrdStatus	Exec Trans Type	Order Qty	Cum Qty	Leaves Qty	Last Shares	Comment
1	New Order(X)					10000				
2		Execution(X)	New	New	New	10000	0	10000	0	
3		Execution(X)	Partial Fill	Partially Filled	New	10000	2000	8000	2000	Execution of 2000
4		Execution(X)	Partial Fill	Partially Filled	New	10000	3000	7000	1000	Execution of 1000
5		Execution(X)	Fill	Filled	New	10000	10000	0	7000	Execution of 7000

New Order and Order Cancel Request Flow

```
8=FIX.4.2 9=0138 35=D 57=Simulator 34=2 49=SLGM0 56=SBI0 52=20070318-
11:16:56 55=MSFT 40=2 38=2000 21=2 11=OrderNumber2 60=20070318-
10:10:10 54=1 44=110.5 10=252

8=FIX.4.2 9=0214 35=8 50=Simulator 34=2 49=SBI0 56=SLGM0 52=20070318-
11:17:00 151=2000 55=MSFT 40=2 11=OrderNumber2 31=0.0 150=0 39=0 54=1
44=110.5 32=0 17=69416515 38=2000 21=2 60=20070318-11:17:00.35 96=0.0
20=0 14=0 37=69399703 10=110

8=FIX.4.2 9=0141 35=F 57=Simulator 34=3 49=SLGM0 56=SBI0 52=20070318-
11:17:02 41=OrderNumber2 55=MSFT 38=2000 11=CancelOrderNumber2
60=20070318-10:10:15 54=1 10=052

8=FIX.4.2 9=0230 35=8 50=Simulator 34=3 49=SBI0 56=SLGM0 52=20070318-
11:17:05 151=2000.0 55=MSFT 11=CancelOrderNumber2 31=0.0 150=6 39=6
54=1 44=110.5 17=69416516 32=0 41=OrderNumber2 38=2000 60=20070318-
11:17:05.35 96=0.0 20=0 14=0.0 37=69399703 10=025

8=FIX.4.2 9=0221 35=8 50=Simulator 34=4 49=SBI0 56=SLGM0 52=20070318-
11:17:05 151=0 55=MSFT 11=CancelOrderNumber2 31=0.0 150=4 39=4 54=1
17=69416517 32=0 41=OrderNumber2 38=2000 60=20070318-11:17:05.35
96=0.0 20=0 14=0.0 37=69399703 10=124
```

Sent Cancel Request – A Closer Look

8=FIX.4.29=014135=F57=Simulator34=349=SLGM056=SBI052=20070318-11:17:0241=OrderNumber255=MSFT38=2000 11=CancelOrderNumber260=20070318-10:10:1554=110=052

35=F	MsgType = F / Order Cancel Request
49=SLGM0	SenderCompID = SLGM0
56=SBI0	TargetCompID = SBI0
55=MSFT	Symbol = MSFT
41=OrderNumber2	OrigClOrdID = OrderNumber2
38=2000	OrderQty = 2000
11=CancelOrderNumber2	ClOrdID = CancelOrderNumber2

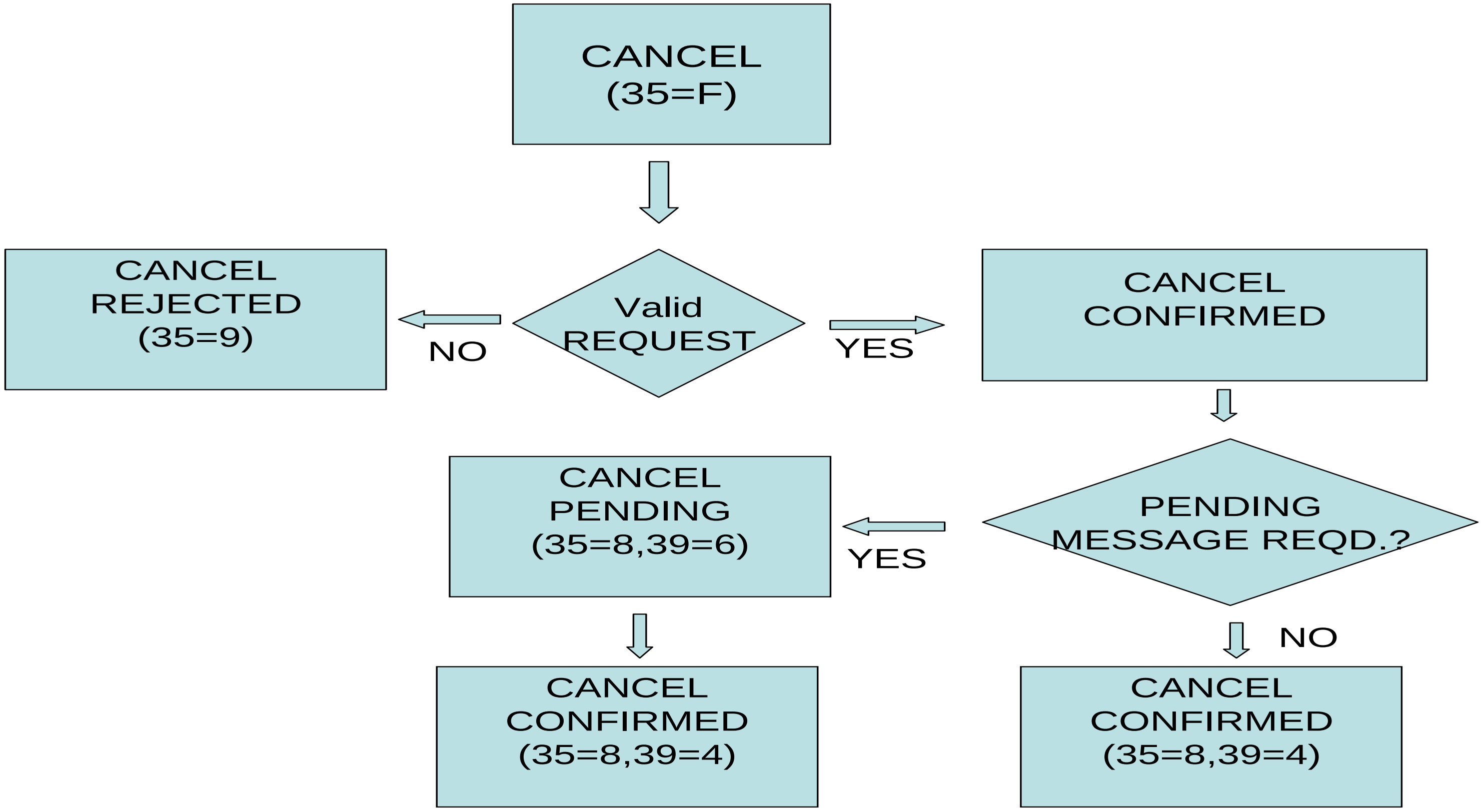
Received Execution Report – Cancel Request Acknowledgement – A Closer Look

35=8	MsgType = 8 / Execution Report
151=2000	LeavesQty = 2000
31=0.0	LastPx = 0.0
150=6	ExecType = 6 / Pending Cancel
39=6	OrdStatus = 6 / Pending Cancel
32=0	LastShares = 0
17=69416516	ExecID = 69416516
6=0.0	AvgPx = 0
20=0	ExecTransType = 0 / New
14=0	CumQty = 0

Received Execution Report – Fully Cancelled – A Closer Look

35=8	MsgType = 8 / Execution Report
151=0	LeavesQty = 0
31=0.0	LastPx = 0.0
150=4	ExecType = 4 / Cancelled
39=4	OrdStatus = 4 / Cancelled
32=0	LastShares = 069416517
17=69416517	ExecID = 69416517
6=0.0	AvgPx = 0
20=0	ExecTransType = 0 / New
14=0	CumQty = 0

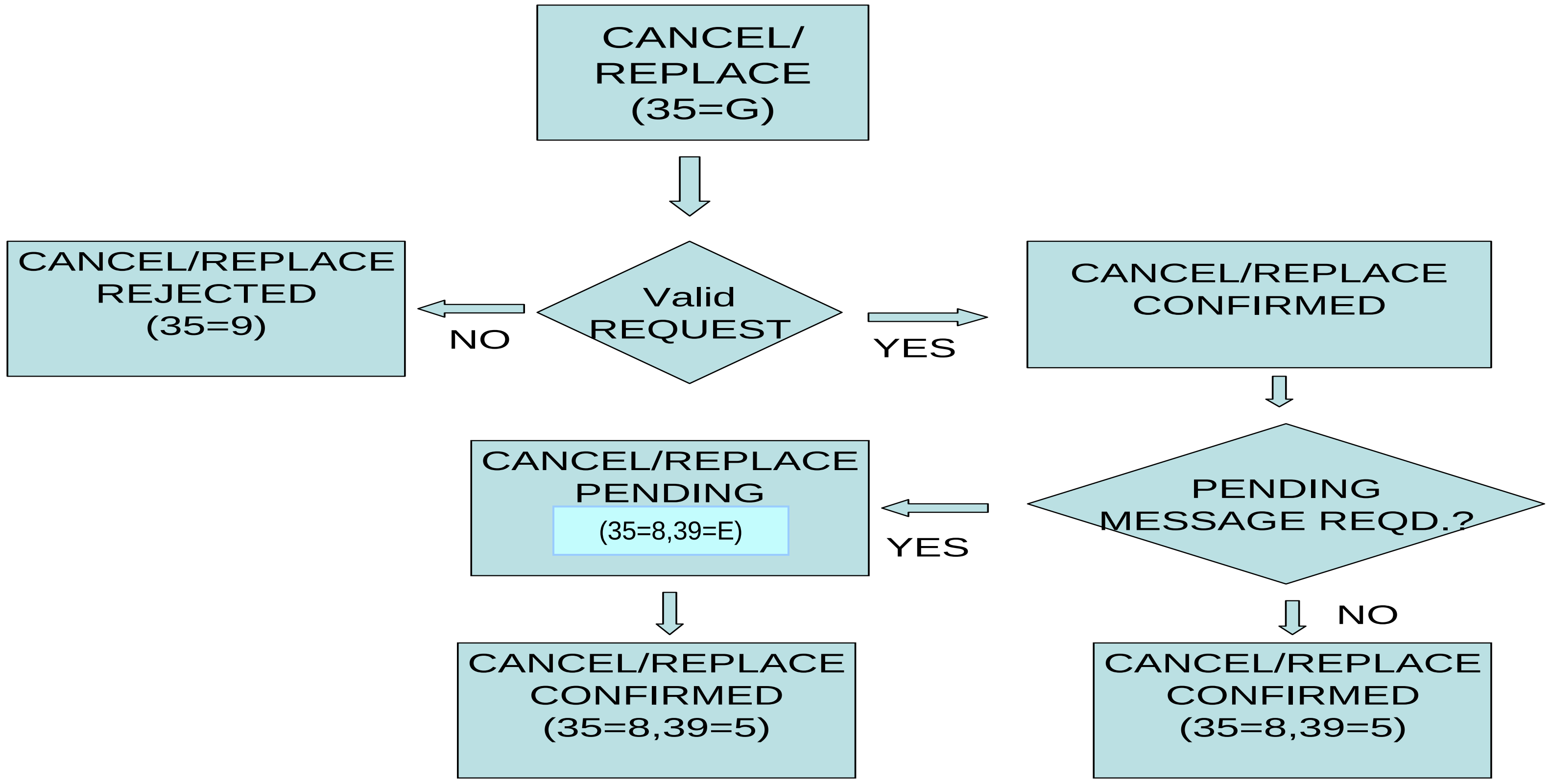
Order Cancel Request



Order Cancel Request Matrix – Zero-filled Order (D3)

Time	Message Received (ClOrdID, OrigClOrdID)	Message Sent (ClOrdID, OrigClOrdID)	Exec Type	Ord Status	Exec Trans Type	Order Qty	Cum Qty	Leaves Qty	Last Shares
1	**New Order(X)**					**10000**			
2		**Execution(X)**	**New**	**New**	**New**	**10000**	**0**	**10000**	**0**
3	**Cancel Request(Y,X)**					**10000**			
4		**Execution (Y,X)**	**Pending Cancel**	**Pending Cancel**	**New**	**10000**	**0**	**10000**	**0**
5		**Execution (Y,X)**	**Canceled**	**Canceled**	**New**	**10000**	**0**	**0**	**0**

Order Cancel/Replace


CANCEL/ REPLACE (35=G)
Valid REQUEST
NO
CANCEL/REPLACE REJECTED (35=9)
YES
CANCEL/REPLACE CONFIRMED
PENDING MESSAGE REQD.?
YES
CANCEL/REPLACE PENDING (35=8,39=E)
CANCEL/REPLACE CONFIRMED (35=8,39=5)
NO
CANCEL/REPLACE CONFIRMED (35=8,39=5)

Order Cancel/Replace – Partial Fill while Pending Replace (D7)

Time	**Message Received** (ClOrdID, OrigClOrdID)	**Message Sent** (ClOrdID, OrigClOrdID)	**Exec Type**	**OrdStatus**	**Exec Trans Type**	**Order Qty**	**Cum Qty**	**Leaves Qty**	**Last Shares**	**Comment**
1	New Order(X)					10000				
2		Execution(X)	New	New	New	10000	0	10000	0	
3		Execution(X)	Partial Fill	Partially Filled	New	10000	1000	9000	1000	Execution for 1000
4	Replace Request(Y,X)					12000				Request increase in order quantity to 12000
5		*Cancel Reject (Y,X)*		*Partially Filled*		*10000*				*If request is rejected*
5		Execution (Y,X)	Pending Replace	Pending Replace	New	10000	1000	9000	0	'Pending replace' order status takes precedence over 'partially filled' order status
6		Execution(X)	Partial Fill	Pending Replace	New	10000	1100	8900	100	Execution for 100 before cancel/replace request is responded to
7		*Cancel Reject (Y,X)*		*Partially Filled*		*10000*				*If request is rejected*
7		Execution (Y,X)	Replace	Partially Filled	New	12000	1100	10900	0	'Partially filled'' order status takes precedence over 'replaced' order status
8		Execution(Y)	Fill	Filled	New	12000	12000	0	10900	Execution for 10900

User-Defined Fields

- If ever you need to add information to a FIX message that does not fit anywhere else, you may employ a User-Defined Field (UDF)

- Tag numbers 5000 through 9999 have been designated user-defined and should not be validated by the FIX engine

- These User-Defined Fields can be 'registered' with FIX Protocol Limited. There are already quite a few UDF listed on the FPL site used by member firms

- Tag numbers 10000 and above have been reserved by FPL for intra-firm communication only and cannot be registered for any specific purpose

User-Defined Messages

- FIX specification conspicuously omits any definition of MsgType ‘U’
- Some FIX engines therefore treat a message of type ‘U’ as a User-Defined Message (UDM)
- Under this convention, MsgType(35) field must begin with ‘U’ to indicate a UDM (for example: “35=U1”)
- Any combinations of fields and values are allowed
- This message is not ordinarily validated by the FIX engine

Review

- Should a FIX engine automatically acknowledge the receipt of an Order message?

  No
- What FIX message is used to change some aspect of an Order?

  Order Cancel/Replace Request (35=G)
- True or False: No tag in a FIX message can be repeated

  False
- True or False: Tag numbers 5000 through 9999 are designated as User-Defined Fields.

  True
- True or False: If a user sees a message with 35=“U” and a number, this is an invalid message type and must be rejected.

  False

Review

- Name three Application message types that must be acknowledged by a trading application

    New Order Single (35=D), Order Cancel Request (35=F), Order Cancel/Replace Request (35=G)

- What, if anything, is wrong with these messages?

    8=FIX.4.2 9=0133 35=D 57=Simulator 34=23 49=SLGM0 56=SBI0 52=20070906-15:45:27 55=IBM 40=2 38=1000 21=2 11=order0 60=20070906-10:10:10 54=1 10=151

    Missing Price(44), which is conditionally required because of OrdType(40) = 2 / Limit

    8=FIX.4.2 9=0210 35=8 50=Simulator 34=32 49=SBI0 56=SLGM0 52=20070906-15:45:29 151=1000 55=NSCP 40=2 11=order2 31=51.24 150=0 39=0 54=1 44=110.5 32=0 17=42327765 38=1000 21=2 60=20070906-15:45:29.828 6=0.0 20=0 14=0 37=41684314 10=131

    This is an order acknowledgement, so LastPx (tag 31) should not be non-zero.

Review

- What, if anything, is wrong with this message?

  8=FIX.4.2 9=0217 35=8 50=Simulator 34=41 49=SBI0 56=SLGM0 52=20070906-15:45:32 151=1000 55=EBAY 40=2 11=order4 31=110.5 150=2 39=2 54=1 44=110.5 32=1000 17=42327797 38=1000 21=2 60=20070906-15:45:32.85 96=110.5 20=0 14=1000 37=41684316 10=219

  LeaveQty(151) should be 0 as OrdStatus(39) and ExecType(150) are Filled(2).

Review

- What, if anything (from an application point-of-view), is wrong with this sequence of FIX messages?

To SBI0:

```
8=FIX.4.2 9=0133 35=D 57=Simulator 34=23 49=SLGM0 56=SBI0
52=20070906-15:45:27 55=IBM 40=2 38=1000 21=2 11=order0 60=20000124-
10:10:10 54=1 44=110.5 10=151
```

From SBI0:

```
8=FIX.4.2 9=0209 35=8 50=Simulator 34=34 49=SBI0 56=SLGM0
52=20070906-15:45:29 151=1000 55=IBM 40=2 11=order0 31=0.0 150=0 39=0
54=1 44=110.5 32=0 17=42327765 38=1000 21=2 60=20070906-15:45:29.828
6=0.0 20=0 14=0 37=41684312 10=045
```

```
8=FIX.4.2 9=0216 35=8 50=Simulator 34=38 49=SBI0 56=SLGM0
52=20070906-15:45:32 151=0 55=IBM 40=2 11=order0 31=110.5 150=2 39=2
54=1 44=110.5 32=1000 17=42327766 38=1000 21=2 60=20070906-
15:45:32.828 6=110.5 20=0 14=1000 37=41684312 10=135
```

Nothing is wrong with this flow

NYSE Technologies℠

NYSE Technologies℠

Powering the exchanging world.℠

Module VII: FIX Routing

Routing Tags in FIX

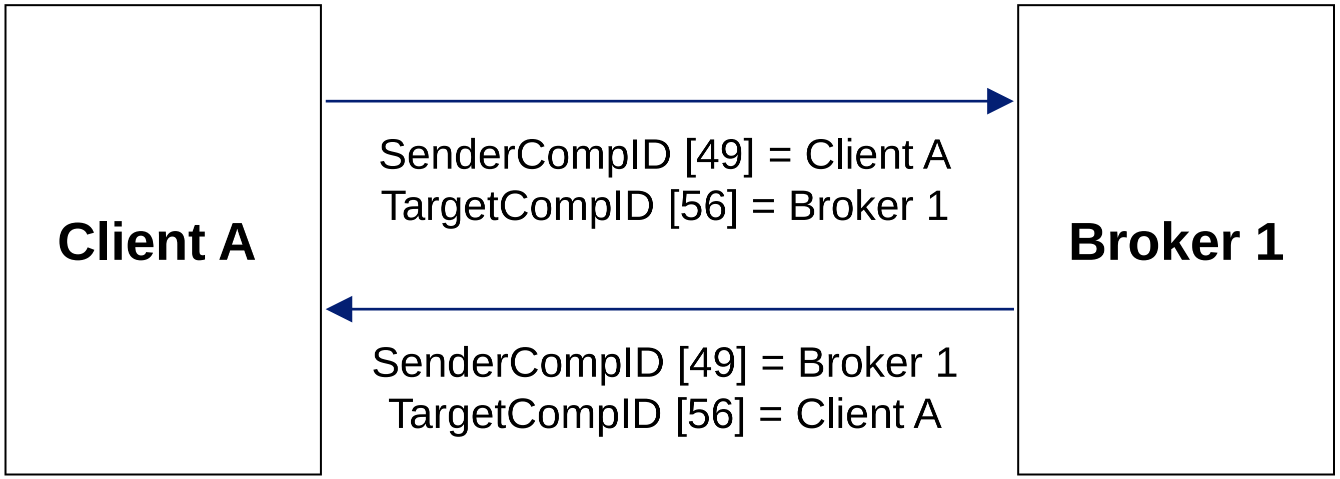
SenderCompID(49) – Identifies the sender of the message – counterparties agree on format

TargetCompID(56) – Identifies the recipient of the message – counterparties agree on format

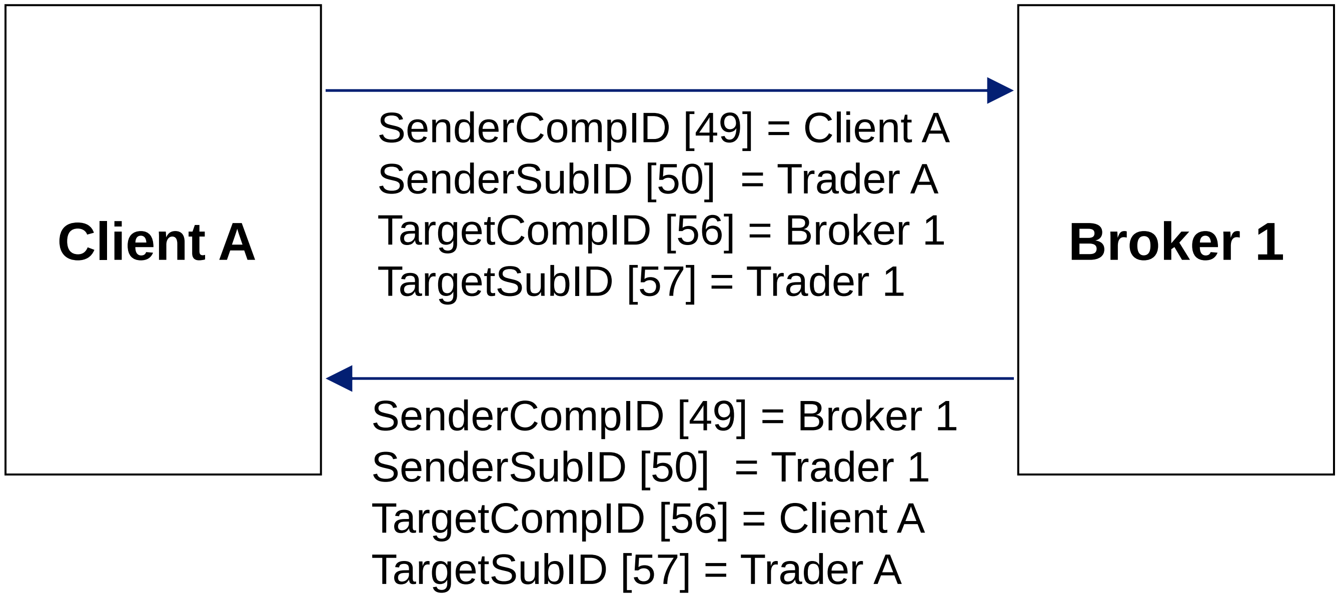
Optional Header fields for routing via third parties

- Hub Routing Tags: DeliverToCompID(128), OnBehalfOfCompID(115)
- Desk & User Location Tags: SenderSubID(50), TargetSubID(57), DeliverToSubID(129), OnBehalfOfSubID(116), SenderLocationID(142), TargetLocationID(143), OnBehalfOfLocationID(144), DeliverToLocationID(145)

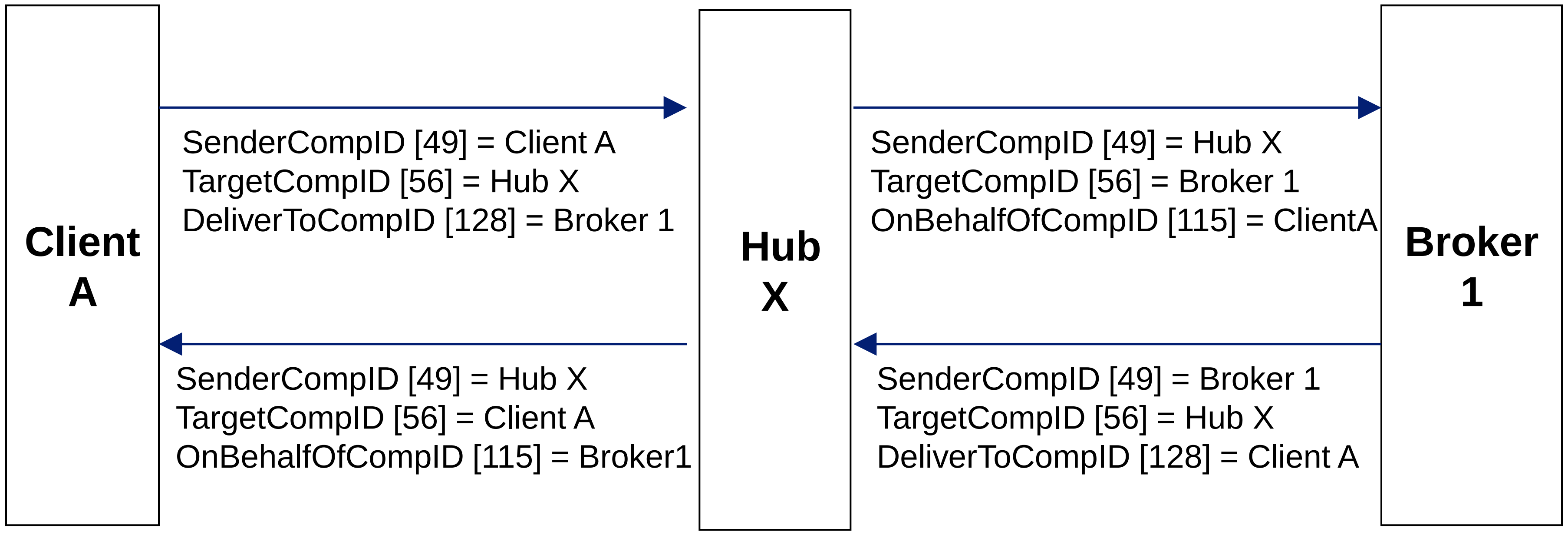
Standard Order Routing Point-to-Point


Client A
SenderCompID [49] = Client A
TargetCompID [56] = Broker 1
SenderCompID [49] = Broker 1
TargetCompID [56] = Client A
Broker 1

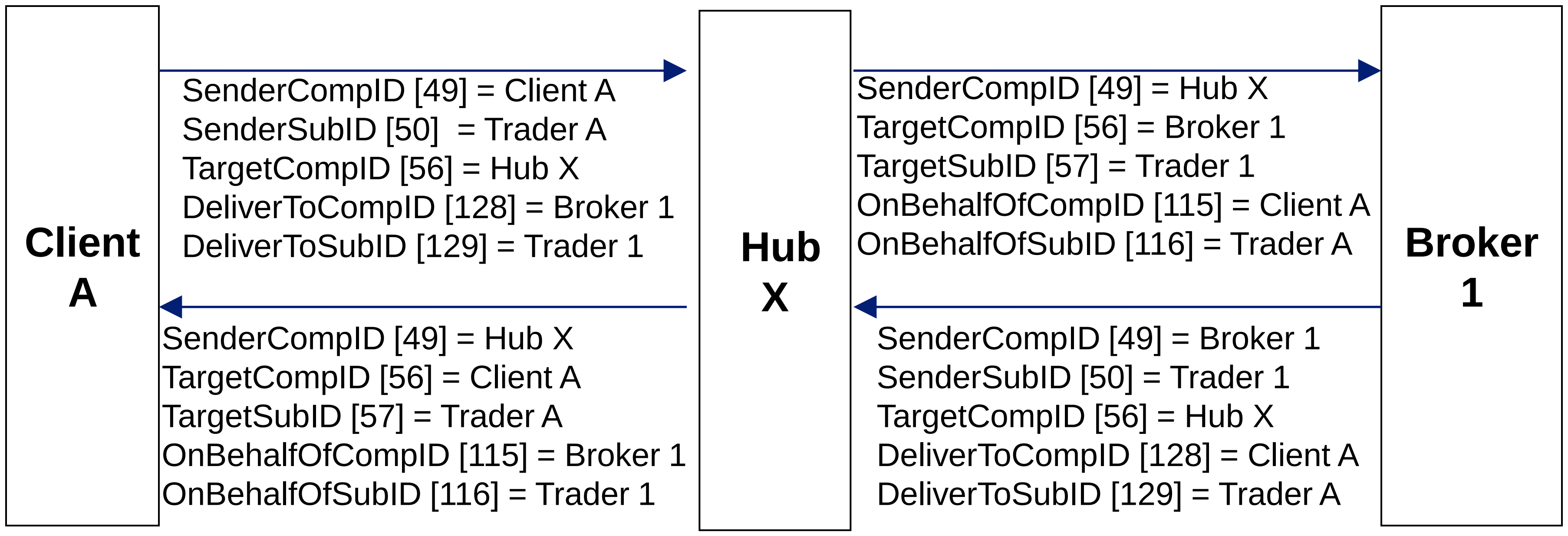
Standard Order Routing Point-to-Point Using Sub-IDs


Client A
SenderCompID [49] = Client A
SenderSubID [50] = Trader A
TargetCompID [56] = Broker 1
TargetSubID [57] = Trader 1
Broker 1
SenderCompID [49] = Broker 1
SenderSubID [50] = Trader 1
TargetCompID [56] = Client A
TargetSubID [57] = Trader A

Standard Order Routing using Third Party as Hub


Client A
Hub X
Broker 1
SenderCompID [49] = Client A
TargetCompID [56] = Hub X
DeliverToCompID [128] = Broker 1
SenderCompID [49] = Hub X
TargetCompID [56] = Broker 1
OnBehalfOfCompID [115] = ClientA
SenderCompID [49] = Hub X
TargetCompID [56] = Client A
OnBehalfOfCompID [115] = Broker1
SenderCompID [49] = Broker 1
TargetCompID [56] = Hub X
DeliverToCompID [128] = Client A

Standard Order Routing using Third Party as Hub Using Sub-IDs


Client A
Hub X
Broker 1
SenderCompID [49] = Client A
SenderSubID [50] = Trader A
TargetCompID [56] = Hub X
DeliverToCompID [128] = Broker 1
DeliverToSubID [129] = Trader 1
SenderCompID [49] = Hub X
TargetCompID [56] = Broker 1
TargetSubID [57] = Trader 1
OnBehalfOfCompID [115] = Client A
OnBehalfOfSubID [116] = Trader A
SenderCompID [49] = Broker 1
SenderSubID [50] = Trader 1
TargetCompID [56] = Hub X
DeliverToCompID [128] = Client A
DeliverToSubID [129] = Trader A
SenderCompID [49] = Hub X
TargetCompID [56] = Client A
TargetSubID [57] = Trader A
OnBehalfOfCompID [115] = Broker 1
OnBehalfOfSubID [116] = Trader 1

Note on FIX Routing

- Important note about routing logic in FIX – it is NOT automatic but rather something that should be handled by a client application or special FIX-to-FIX routing engine
- For example, if my FIX engine received a message with DeliverToCompID set, it may not automatically forward that message to the firm specified in that field
- Routing is still up to the application, which may check an incoming message to see whether DeliverToCompID is set, and if desired, take the message and route it to firm specified

NYSE Technologies℠

Powering the exchanging world.℠

Module VIII: Integrating FIX Into Your Electronic Trading Infrastructure

Integrating FIX into a Trading Architecture

Considerations when adding FIX to a trading architecture

- Application integration
- Integration steps
- Network Options
- FIX Engine – Buy versus Build

FIX Engine – Integration Process

Integration steps

- Determine trading requirements (what kind of trading will be done)
- Evaluate counterparties and their ability to use FIX
- Select FIX engine (Buy vs. Build)
- Select middleware (how application will interface with FIX engine)

FIX Engine – Integration Process

Integration steps

- Develop trading application (if not using pre-built OMS)
- Testing
  - internally (trade simulators)
  - counterparties
- Train users and operators – support staff

FIX Engines – Ease of Use and Integration

Application Integration

- Difficulty depends on choice of using pre-certified OMS versus internally-developed trading application
- Choice of integration via some middleware
  - Tibco Rendezvous
  - IBM MQ Series / WebsphereMQ
  - JMS
  - Point-to-point over TCP/IP socket

FIX Engines – Ease of Use and Integration

Database Integration

- FIX engine database
  - Type of database supported by FIX Engine for persistence
    - Flat-file
    - RDBMS
- Application database
  - Relational or flat-file database used by trading application

FIX Engines – Buy vs. Build

Buy – Advantages

- Vendor maintains responsibility for FIX engines
- New version of FIX? Upgrade done by vendor
- Staff can better focus on the trading systems and business
- Decreases FIX engine testing time (as FIX engine testing is done by vendor)
- Vendor FIX engine already production-tested with many counterparties worldwide
- Vendor provides support (both integration and production)

FIX Engines – Buy vs. Build

Buy – Disadvantages

- Capital cost (license fees, maintenance fees)
- Additional cost for any custom work
- Development and capability of FIX engine is out of customer’s hands
- Support considerations (must rely on vendor for support, not in-house support)
- If you only need to connect to one counterparty, or have low volume, a vendor FIX engine may have more functionality than you need

FIX Engines – Buy vs. Build

Build – Advantages

- No reliance on third-party vendor as user maintains responsibility for FIX engine – stays ‘in-house’
- Can tailor FIX engine to business requirements (might not need to support all message types)
- Possible cost savings if built by in-house development staff

FIX Engines – Buy vs. Build

Build – Disadvantages

- User is responsible for development, support, and maintenance of FIX engine
- Extended test cycle – need to test all FIX engine functionality
- Time-consuming – development cycle is likely to be a minimum of one year, depending on requirements
- Any new FIX version – additional internal development effort

Network Options

Choosing a network and connection model is a critical component of building an electronic trading infrastructure

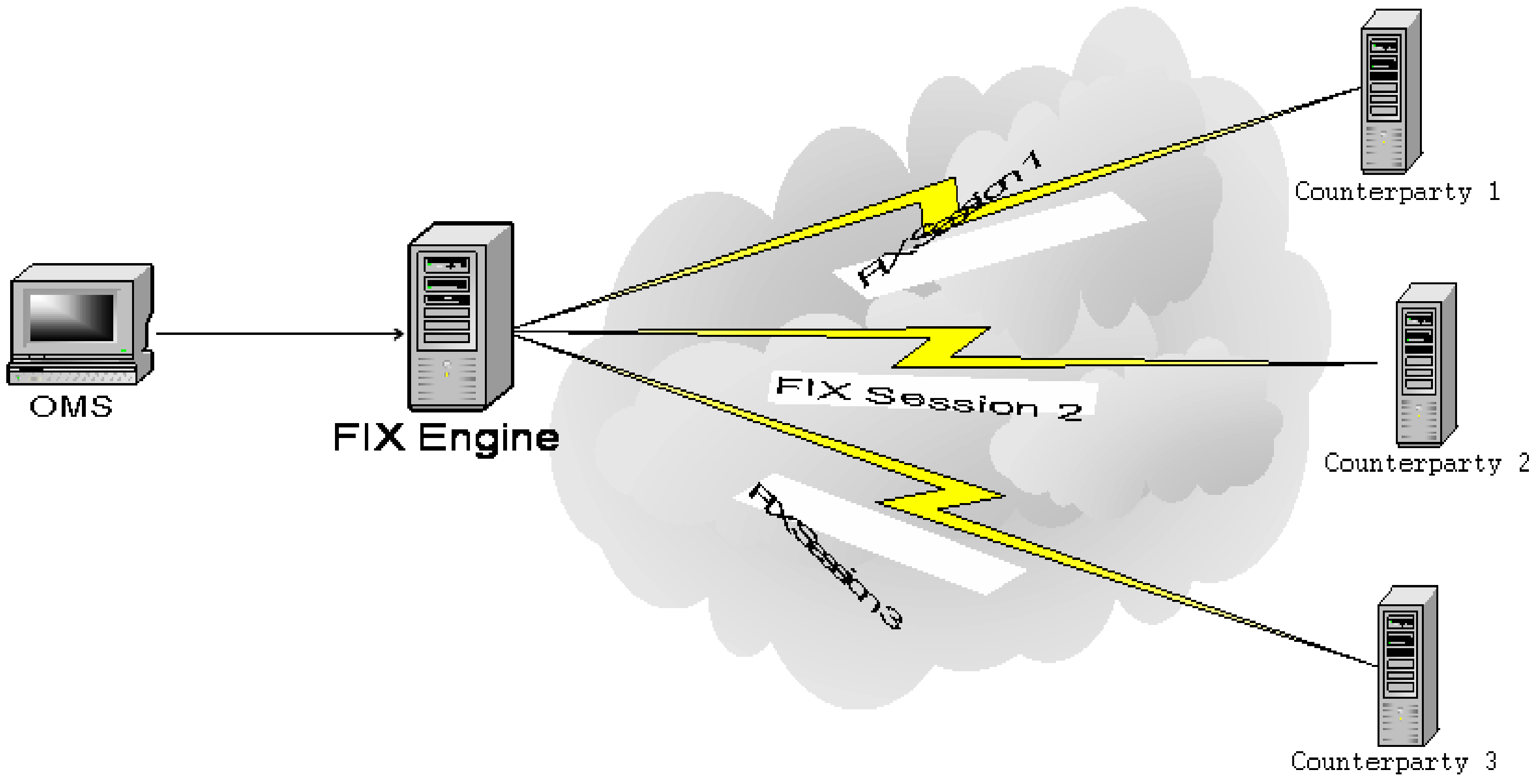
Point to Point Connectivity

- Leased Line / Circuit
- VPN / IP Tunnel
- Internet (with / without FIX encryption)

Third Party / Indirect Connectivity

- Hub-and-Spoke Network

Point-to-Point FIX Sessions



Point-to-Point Connections

- Counterparties connect directly to each other – either through leased lines or virtual private network
- FIX operation, monitoring, and connection maintenance is the responsibility of connecting parties
- If a firm has multiple counterparties, it will have to manage multiple connections and certify each of them

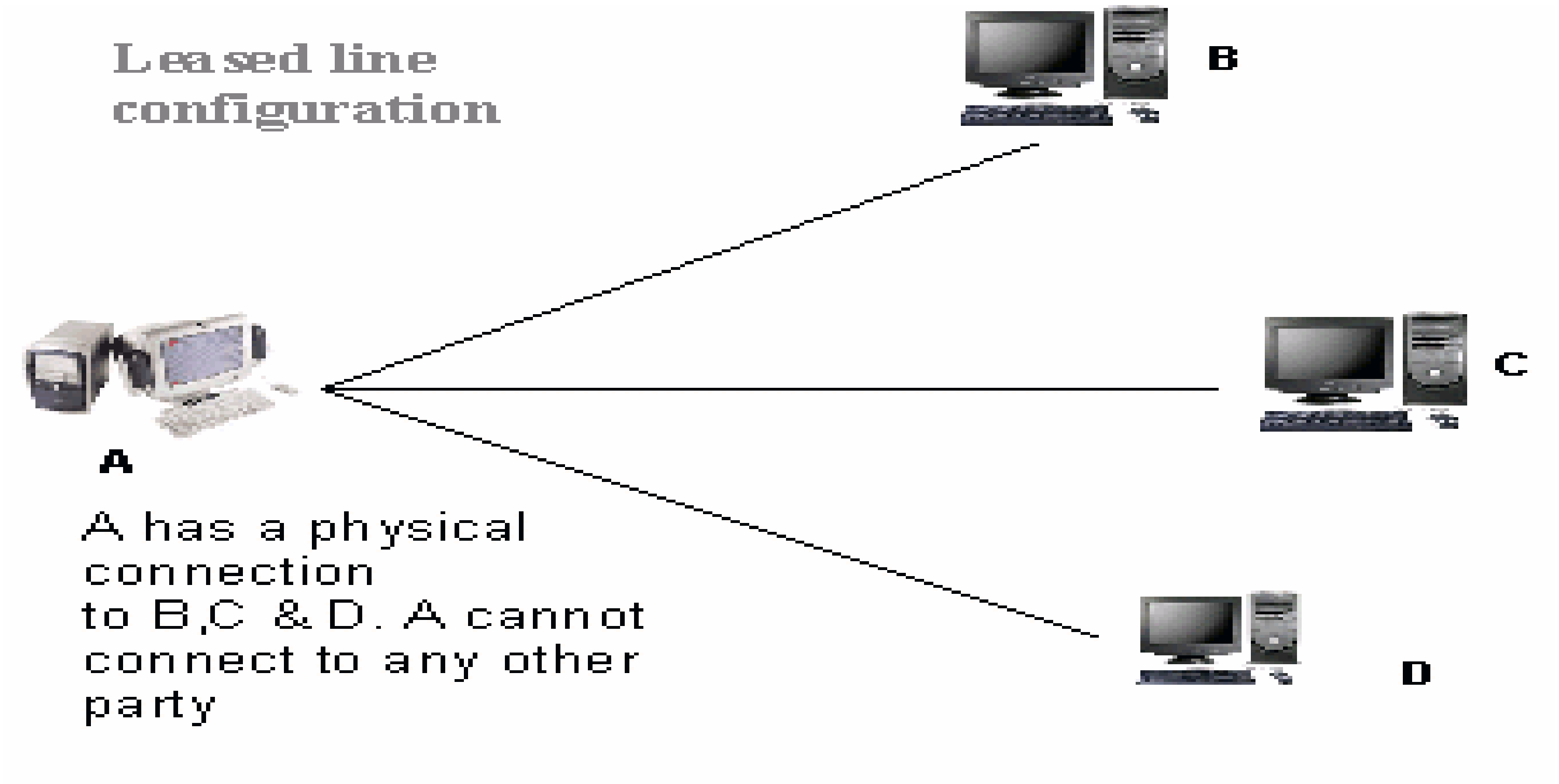
Pros

- Connections are private, messages are untouched

Cons

- Greater overhead due to the on-boarding and certification required for each connection

Point-to-Point Over Leased Lines



A has a physical connection to B,C & D. A cannot connect to any other party

Point-to-Point over Leased Lines

- Direct "point-to-point" connection between you and your counterparty via telecommunications circuit

- Fixed cost depending on size of the circuit (i.e. 56k, 64k, 128k, T1, T3 lines)

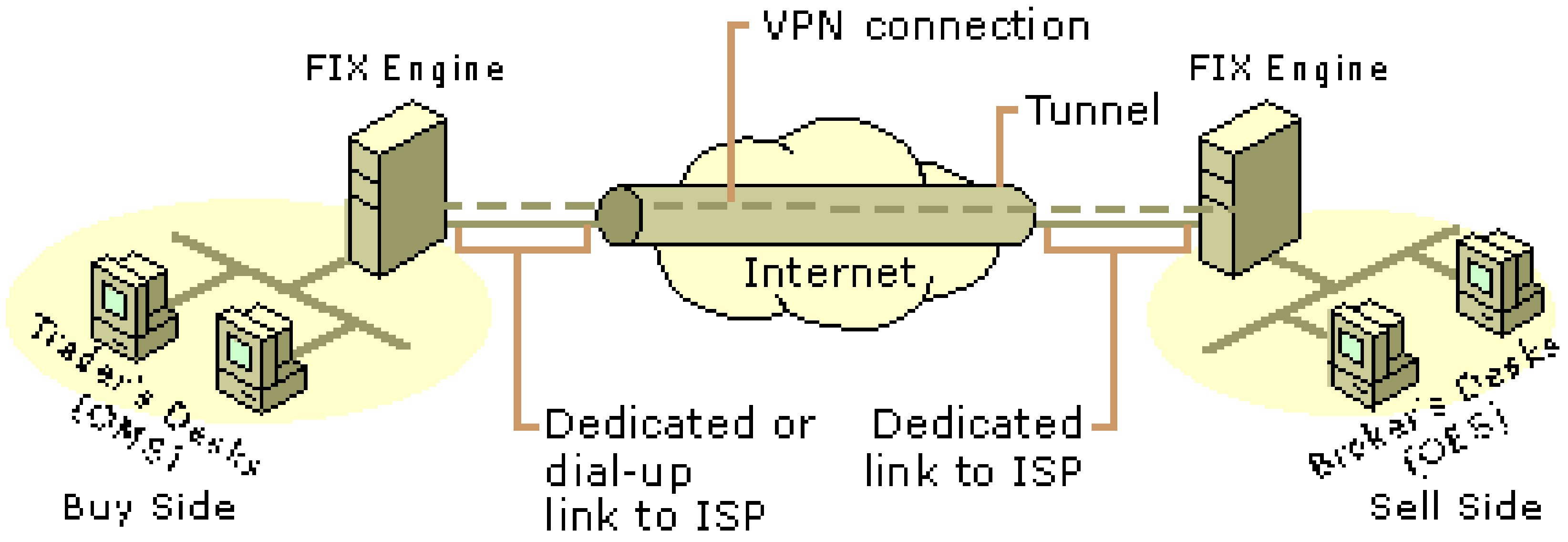
Pros

- Control on throughput
- Can be always on
- Private physical connection between trading partners

Cons

- Costly for multiple connections, especially if cross-boundary
- Time to market – can take several weeks to months for installation

Point-to-Point Over Virtual Private Network (VPN)



Point-to-Point Over Virtual Private Network (VPN)

- One physical internet connection and separate logical connections to each of your counterparties
- FIX engines do not connect directly. Connectivity is via networking device (e.g. router) that securely tunnels through the Internet

Pros

- Provides security through encryption
- Reduced cost and complexity for several connections
- You manage a single physical line into the network. Adding connections is straightforward

Cons

- Depends on reliability of the Internet
- Increased risk: If you lose your connection to the network, you lose all connections
- No network monitoring from networks like NYSE, TNS, etc.

Point-to-Point over Internet

- Provides low-speed connection at little cost or higher-speed connection at a greater cost.
- Similar to VPN connection but without encrypted tunnel
- Not many businesses use due to security risk involved

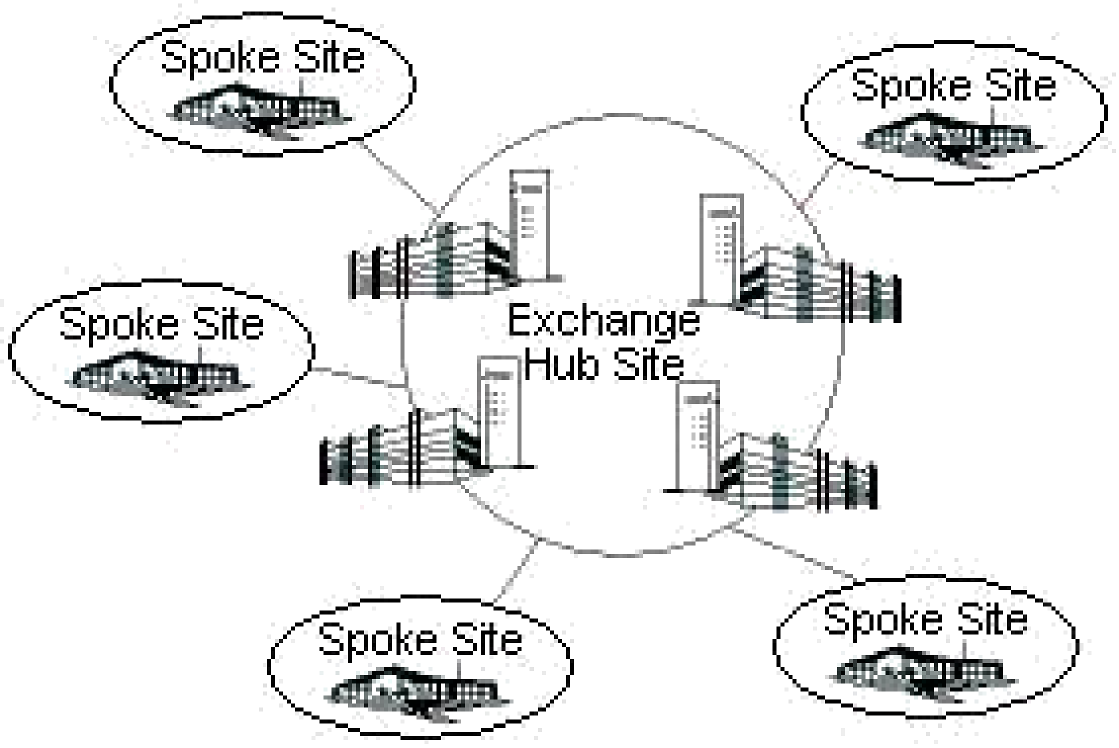
Pros

- Significant cost savings
- Does not require dedicated hardware resulting in faster set up

Cons

- Insecure
- Less reliable
- Higher risk of connection failure and message interception

Hub-and-Spoke Network



Hub-and-Spoke Network

- All communication between you and trading partners passes through a hub
- Firms (spoke sites) maintain one connection to the hub site, through which they communicate with any other participant on the network
- Hub provides value added services such as routing, monitoring, message normalization and translation, validation, and certification

Pros

- Ability to outsource FIX connection management
- Certify once to the hub site. Hub maintains certifications to other counterparties
- Minimal effort to add new connections

Cons

- Performance – the hub can become a bottleneck
- Risk – the hub can be a single point of failure

Things to Consider in Hub Selection

Community

- Availability of counterparties you need to connect to

Cost

- Identify the cost model for each option: infrastructure cost, installation cost, usage cost (i.e. per message, per bandwidth size, percentage of trade value)

Service

- Uptime/availability, reliability, and support model (i.e. any SLA)

Redundancy

- Backup connectivity and business continuity plan

Time to market

Business Continuity / Disaster Recovery

Ability to continue trading upon outage

Includes:

- Replication network
- Replication of hardware
- Replication of software
- Replication of data
- Automatic or manual failover from primary to backup server(s), networks

Business Continuity / Disaster Recovery

Benefits of Automatic Failover

- Minimal downtime (usually 99.99% availability)
- Replication of data to local and remote backup servers
- Seamless connectivity for applications and counterparties
- Protection against hardware and operating system failures and network outages

Disaster Recovery in the FIX Engine

- Important to have local as well as remote backup

- Backups should be updated in real-time (HA)

- Recovering FIX environment (locally)
  - If local backup, backup should become primary automatically
  - FIX engine should take care of requesting lost messages from counterparties
  - Application may need to be reconnected

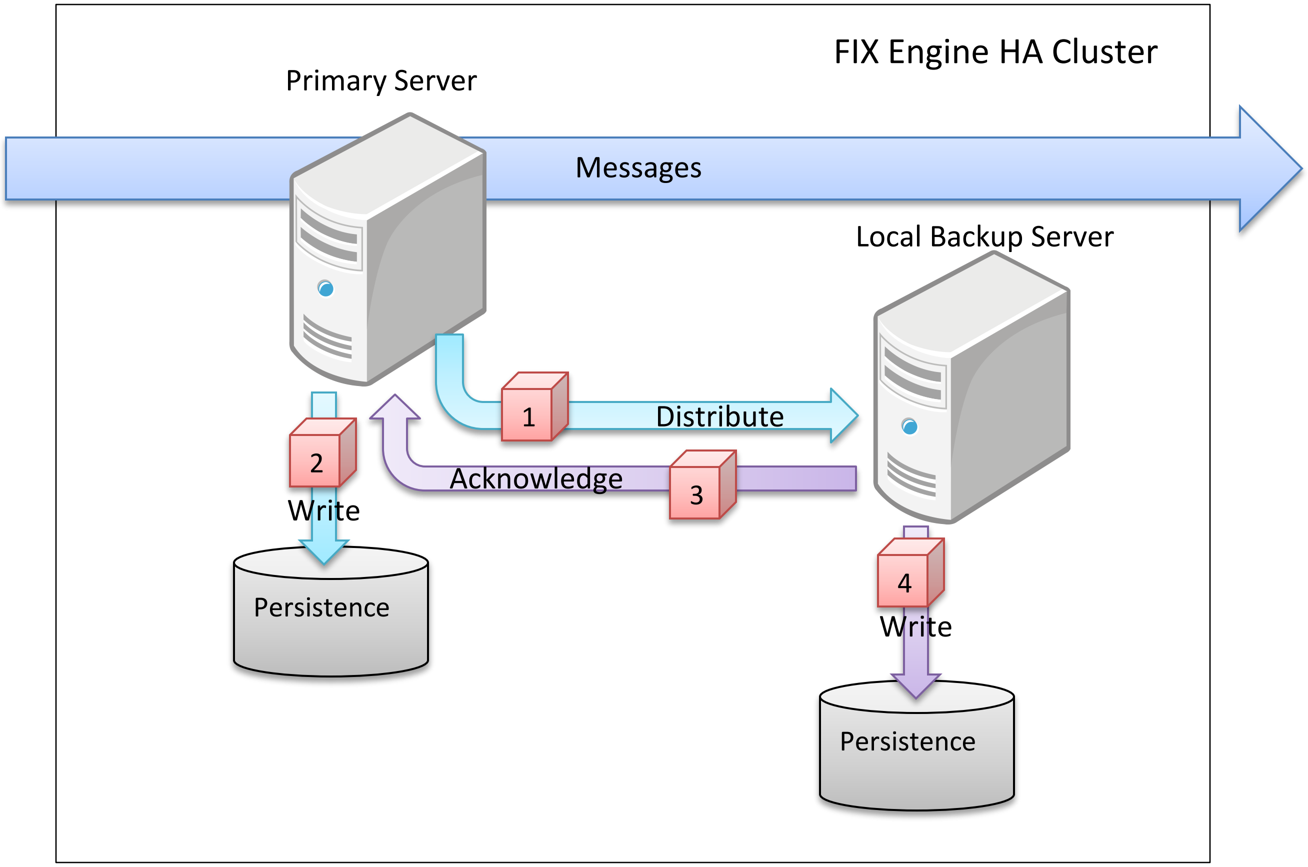
Disaster Recovery in the FIX Engine

Recovering FIX environment (from remote site)

- Notify the trading desk and other business stakeholders to switch to manual mode
- Notify counterparties
- Connect trading application to remote site FIX engine
- Activate remote site and connect FIX sessions
- Recover the missed messages from failed engine’s log file

FIX Engine HA with Synchronization (LAN)

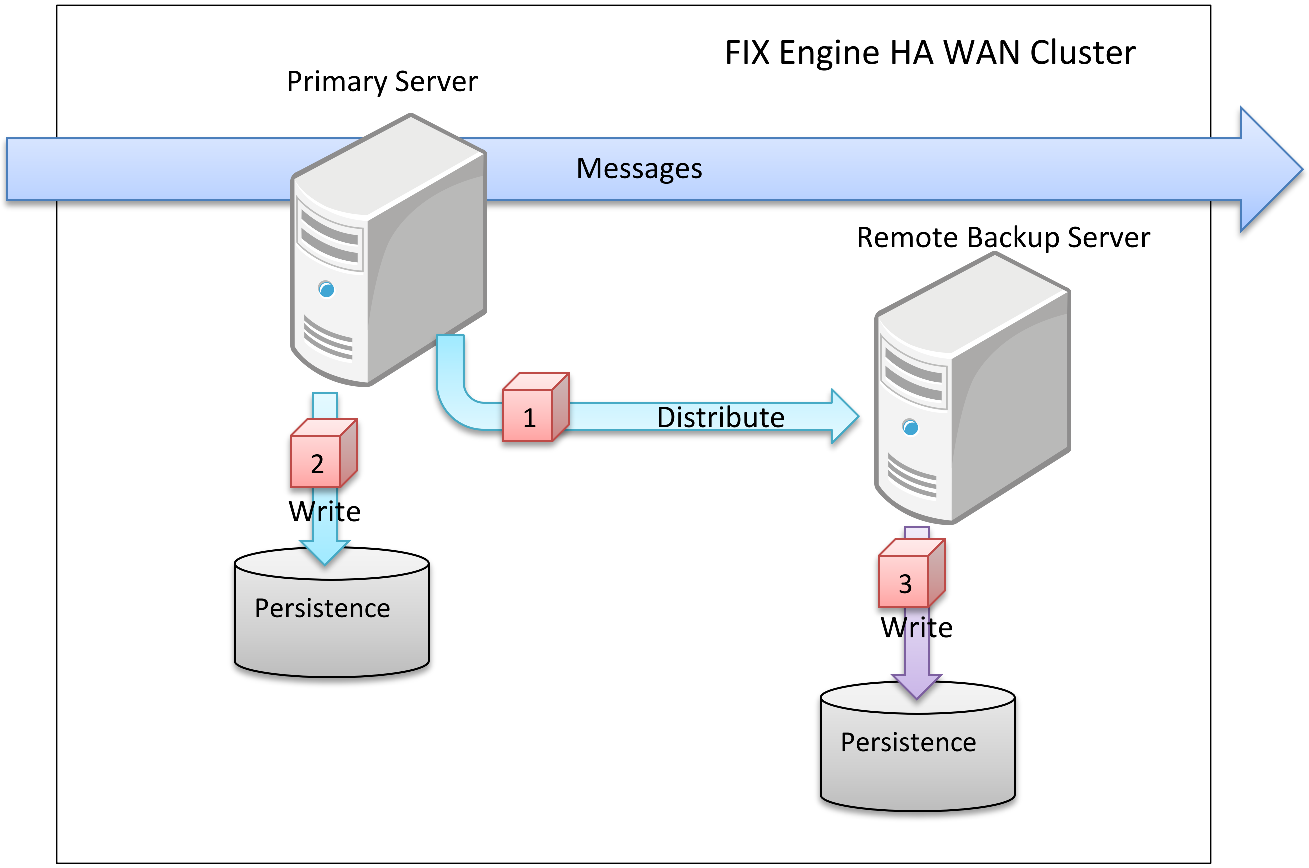
Primary server distributes all FIX messages, session state and configuration data to secondary in real-time. Secondary saves that data in its database and acknowledges back to primary



NYSE Technologies

FIX Engine HA with Replication (WAN)

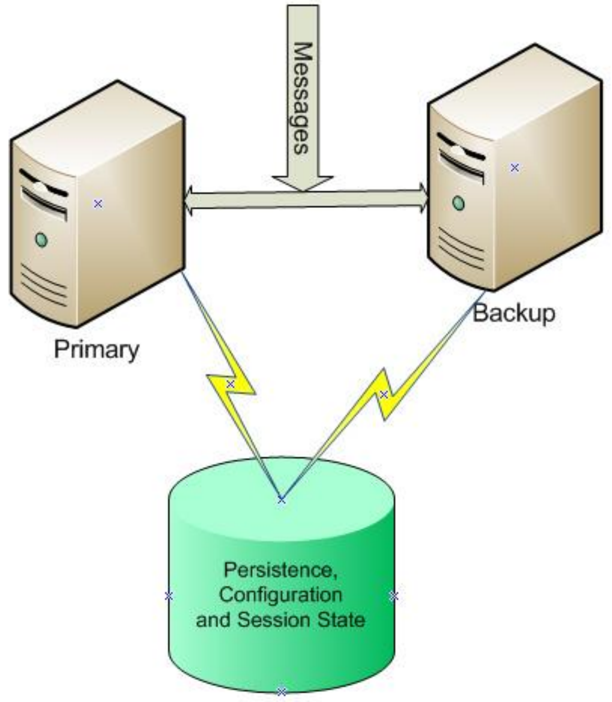
Primary server distributes all FIX messages, session state and configuration data to secondary in real-time. Secondary saves that data in its database but does not acknowledge back to primary



FIX Engine Non-replicated HA

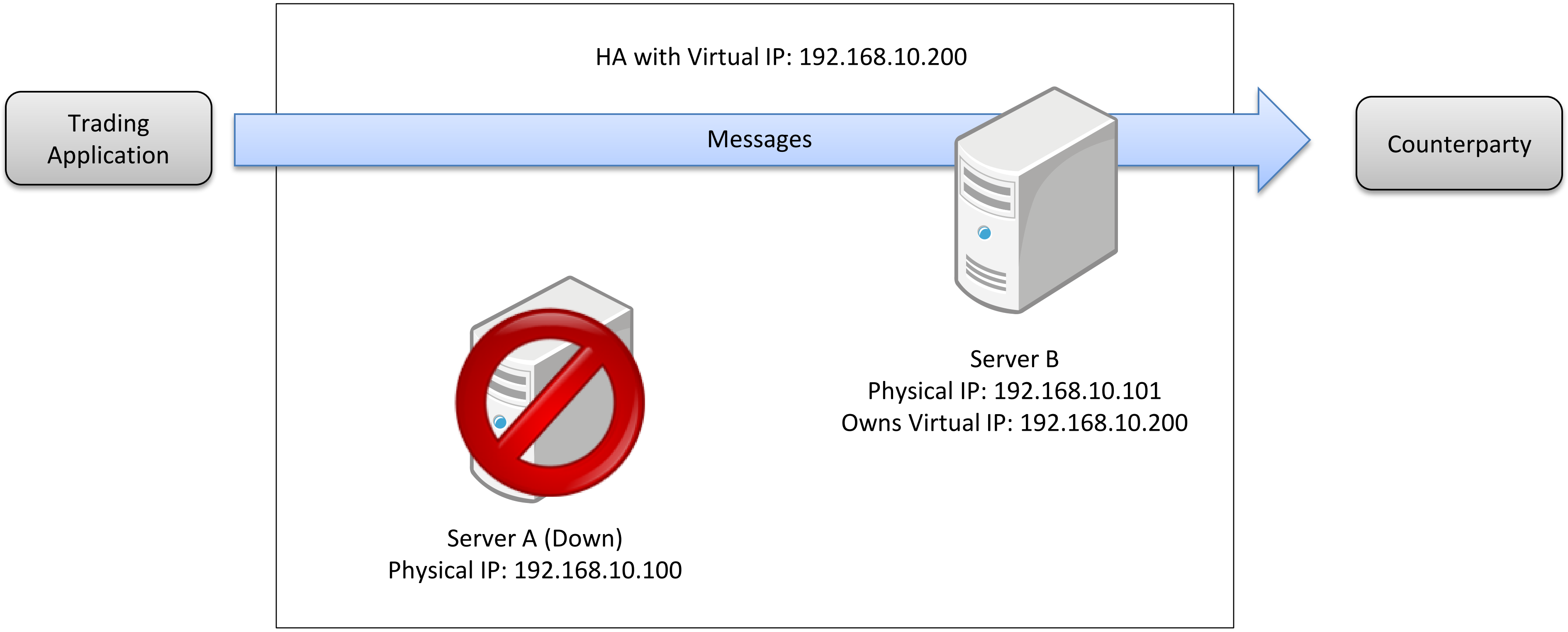
In non-replicated HA, the configuration, sessions state and persistence data is stored on a highly available network drive (NAS etc.) and it shared by all servers in the node

Since no replication takes place, there is improved performance on the active node

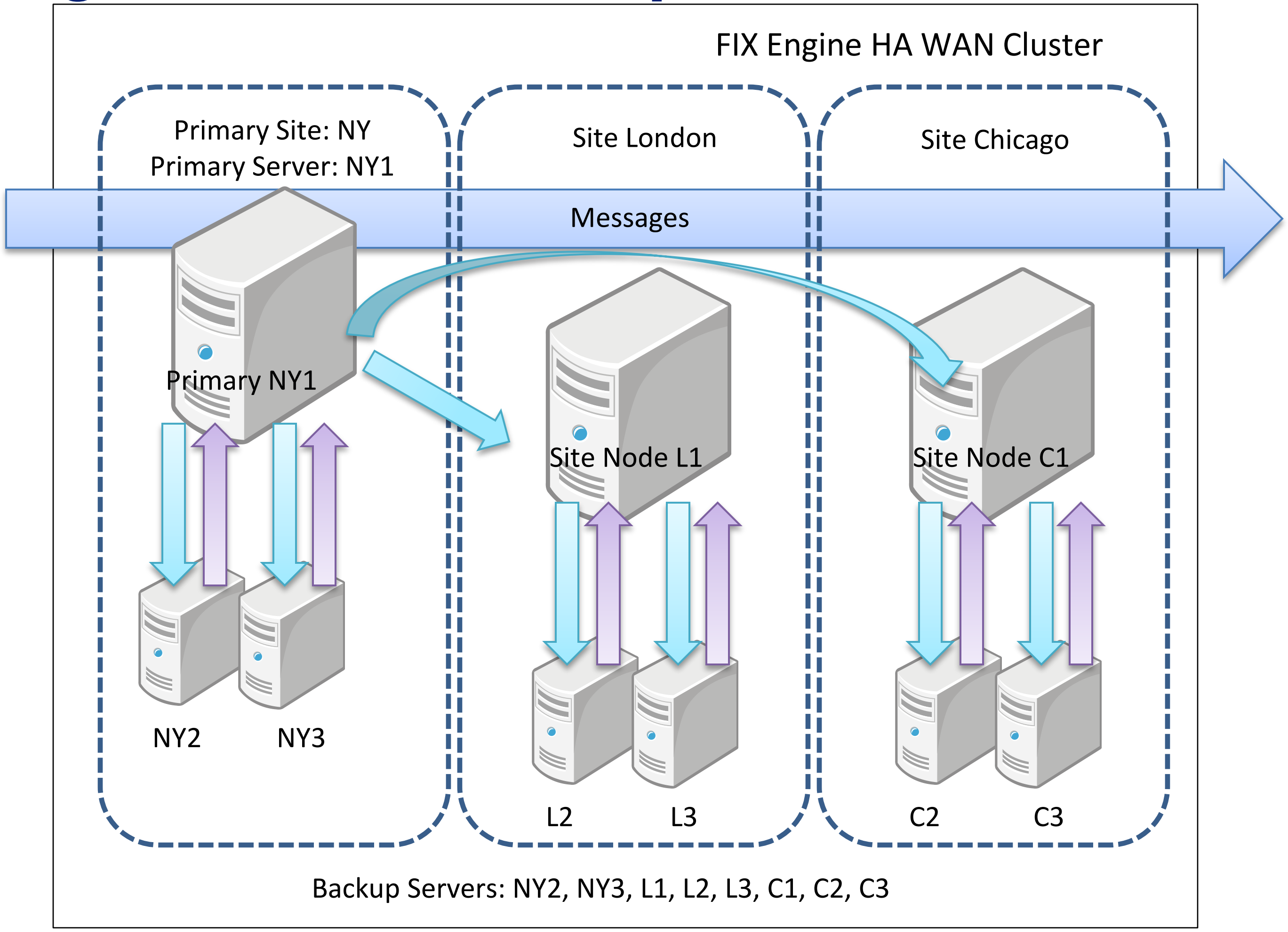


FIX Engine HA Failover

If the primary goes down, secondary will take up all sessions and continue where the primary left. If the primary comes back online, it will be become the secondary


HA with Virtual IP: 192.168.10.200
Trading Application
Messages
Counterparty
Server B
Physical IP: 192.168.10.101
Owns Virtual IP: 192.168.10.200
Server A (Down)
Physical IP: 192.168.10.100

FIX Engine Multi-Site Replication



Local synchronization and failover
Site replication and failover

NYSE Technologies℠

Powering the exchanging world.℠

Module IX: Rules of Engagement

Rules of Engagement – What is it?

- The mutual contract of business rules and operational practices that governs the FIX session between two parties.

- The Rules of Engagement supplement the official FIX Protocol specification and include written and implied practices specific to the two parties.

Rules of Engagement – Session

Session-oriented rules may include

- Escalation and Support contacts
- Network connection configuration (IP addresses, ports)
- FIX versions supported
- Encryption support (i.e. PGP-DES-MD5)
- Counterparty identification
- Start-of-Day / End-of-Day procedures
- PossDup and PossResend handling
- Failover procedures
- Test scripts

Rules of Engagement – Business

Business-oriented rules may include

- Times of operation
- Message types supported
- Markets and asset classes supported
- Order and settlement Currency
- Security identification conventions
- Order state change and ID chaining conventions
- Handling of duplicate messages or unsolicited modifications
- Routing rules and determination
- Order open and expiry rules

Rules of Engagement – Counterparty ID

- Counterparty identifiers (CompID) can be anything (a few letters is typical)
- Both parties must agree how they identify each other
- Which tags are used and which identifier in each depends on the type of network connecting the parties
  - The Company ID tags SenderCompID(49) and TargetCompID(56) are required fields and always used regardless of context
  - Hub routing tags DeliverToCompID(128) and OnBehalfOfCompID(115) are optional may be used in hub-and-spoke network configurations
  - The hub also may make a routing decision based on any other information, including the optional User & Desk routing tags SenderSubID(50), TargetSubID(57), DeliverToSubID(129), and OnBehalfOfSubID(116)

Start-Of-Day / End-Of-Day Procedures

- Defines FIX session schedule
    - Logon
    - Logout
    - **E**nd-**O**f-**D**ay (EOD)

- Who will initiate the session at start time?

- Minimum down time required to complete EOD?

- Can EOD be performed while connected (35=A, 141=Y, 34=1)?

Rules of Engagement – Symbology

- Naming convention to be used to **identify securities** in a FIX message (ticker, ISIN, RIC, SEDOL, CUSIP, etc.)
- Securities may be **listed in more than one location** – Ticker is not always suitable for international trading
- Fields in FIX messages used for identification of securities
  - *Symbol(55)* – typically local exchange ticker
  - *IDSource(22)* – i.e. CUSIP, ISIN, RIC, SEDOL
  - *SecurityID(48)* – value for *IDSource* above (i.e. RIC of “IBM.N”)`

Rules of Engagement – Business Messages

- Supported FIX versions & messages in each version
  - Asset classes
  - Order types – i.e. Single / List / Multi-leg, Day / Multi-day, Stop, etc.
  - Other messages – i.e. IOIs/ATs, Allocations / Confirmations, etc.
  - Order expiry: Day, Multi-day (i.e. GTC), IOC, etc.
- Values supported by various FIX tags
  - Requirements for specific field values, and any modifications or customizations to FIX specifications
- Attributes of an order allowed to amend
- Number of decimals values like average price for rounding
- Support for Execution Cancel (Bust) and Correction, Done for Day, Status, Don’t Know Trade, etc.

Rules of Engagement – Test Scripts

- Test scripts are critical to ensure safe and continuous operation of the system
- Specific test scripts to be executed by a newly-connected counterparty are highly recommended
- This section would describe the test environment and any specifics that would help quickly implement the testing effort
- The FIX Protocol specifications contain an extensive amount of testing material
  - Session Level Test Scenarios – Volume 2
  - Order State Change Matrices scenarios (formerly known as “Appendix D”) in Volume 4 – Fix Application Messages: Orders And Executions (Trade)

Client On-Boarding – What's Involved?

- Establish Business / Technical Contacts, Escalation Points
- Establish Rules of Engagement
- Test Session build
- Session, Application, User Testing
- Production Session Build
- Production Testing
- Rolling Into Production

Testing – What's Involved?

Development Environment

- Test development and application integration

UAT Environment

- Involve / train actual users and counterparties

Production Environment

- Market trial, pre go-live testing

Testing – What's Involved?

Testing should cover all the possible scenarios

- Session-level testing
- Application-level testing
- Process changes
- Fall-back procedures
- Where possible, perform full end-to-end tests

Test all system linkages

- Test the whole process end-to-end

Stress/volume tests

- Make sure your system can handle high volumes
- Make sure your FIX engine can handle 'stressful' scenarios (e.g. high volume message resend requests)

What should be tested?

Basic connectivity / integration testing

- Establish IP connectivity
- Ensure FIX engines, applications and databases talk to each other

Session-level testing

- FIX Session Level Test Scenarios (Volume 2)

Business / application-level testing (manual or automated tool)

- Order State Change scenarios (formerly Appendix D)
- Client implementation specifics
- Volume Testing / Requirements

Pre Go-live / Market trial

- Controlled first orders / counterparties

Failover / Backup Testing

Test the procedure and ease of falling back to non-electronic procedures if FIX connectivity goes down for long period of time

- Back-up trading platforms
- Telephone orders, amends, cancels, etc.
- Ensure you can get your systems back in sync when everything starts working again
- Notifying your clients and brokers
- Notifying any other interested parties
  - Networks
  - Clearing agencies
  - Settlement agencies

Failover / Backup Testing

How do you find out if something's gone wrong?

- System diagnostics and monitoring
- FIX connection monitoring
- For broker or vendor solutions, the broker/vendor should be monitoring the system for you

How do you recover?

Phased Implementation Options

- Phase by message type, e.g. IOIs, then Orders and Execution Reports, then List Orders, etc.

- Phase by market, e.g. first Hong Kong, then London, then New York

- Phase by counterparty connections

- Phasing allows new systems and/or processes to ‘bed in’ in a relatively low risk, though time-consuming, manner…

Topics for care and caution

- Counterparties behave differently
- Verify routing to correct desks/counterparties, internal subsystems, or desks/traders
- Overnight order handling, e.g. cross-continent trading
- Timing of Order Acks & Fills
- Session availability and timing
- Order expiration: Day / GTC / DFD
- Symbol identification
- Cross-country currency confusion
- Cancel and Cancel/Replace functionality

NYSE Technologies℠

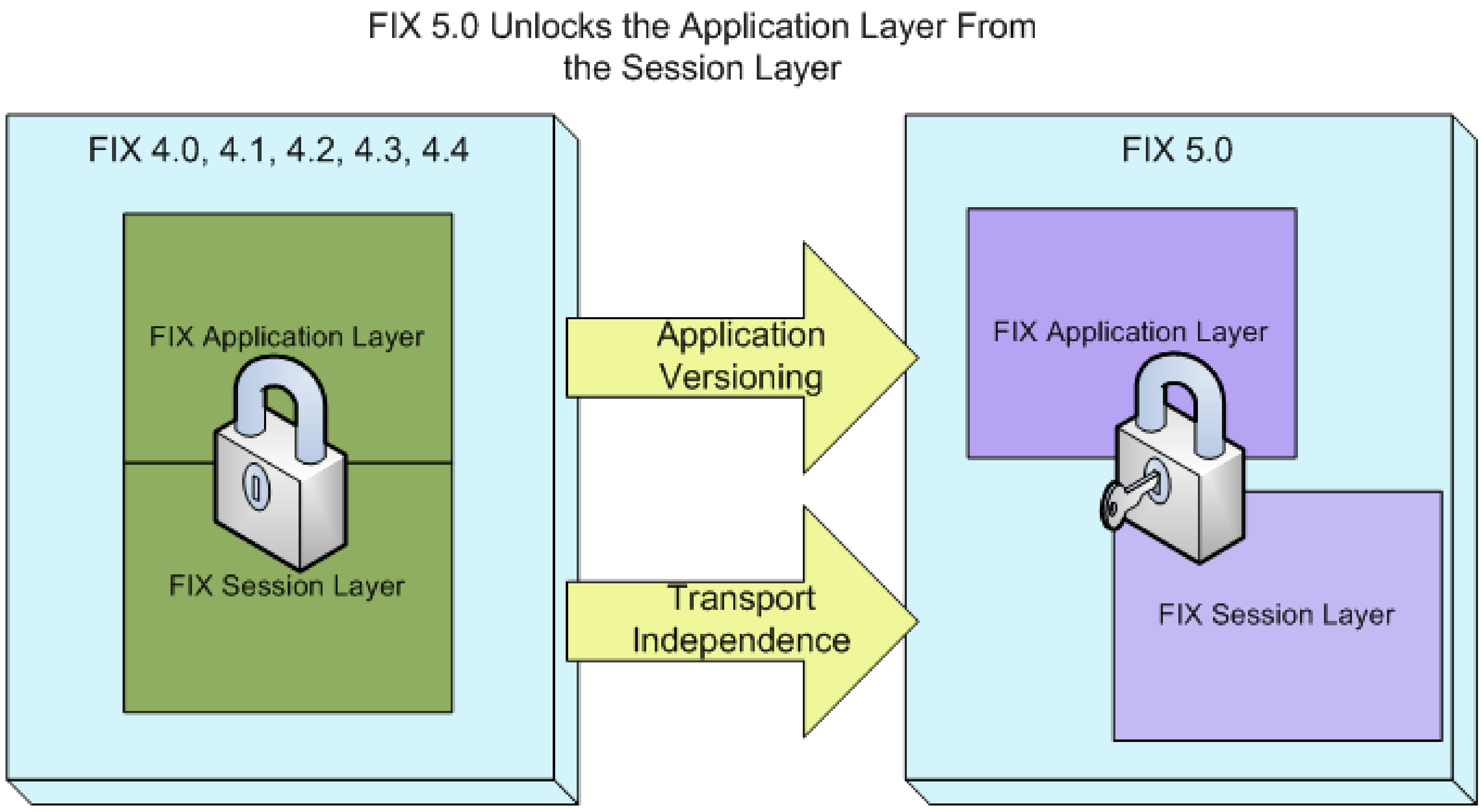
Powering the exchanging world.℠

Module X: What's New in FIX?

What’s New in FIX 5.0?

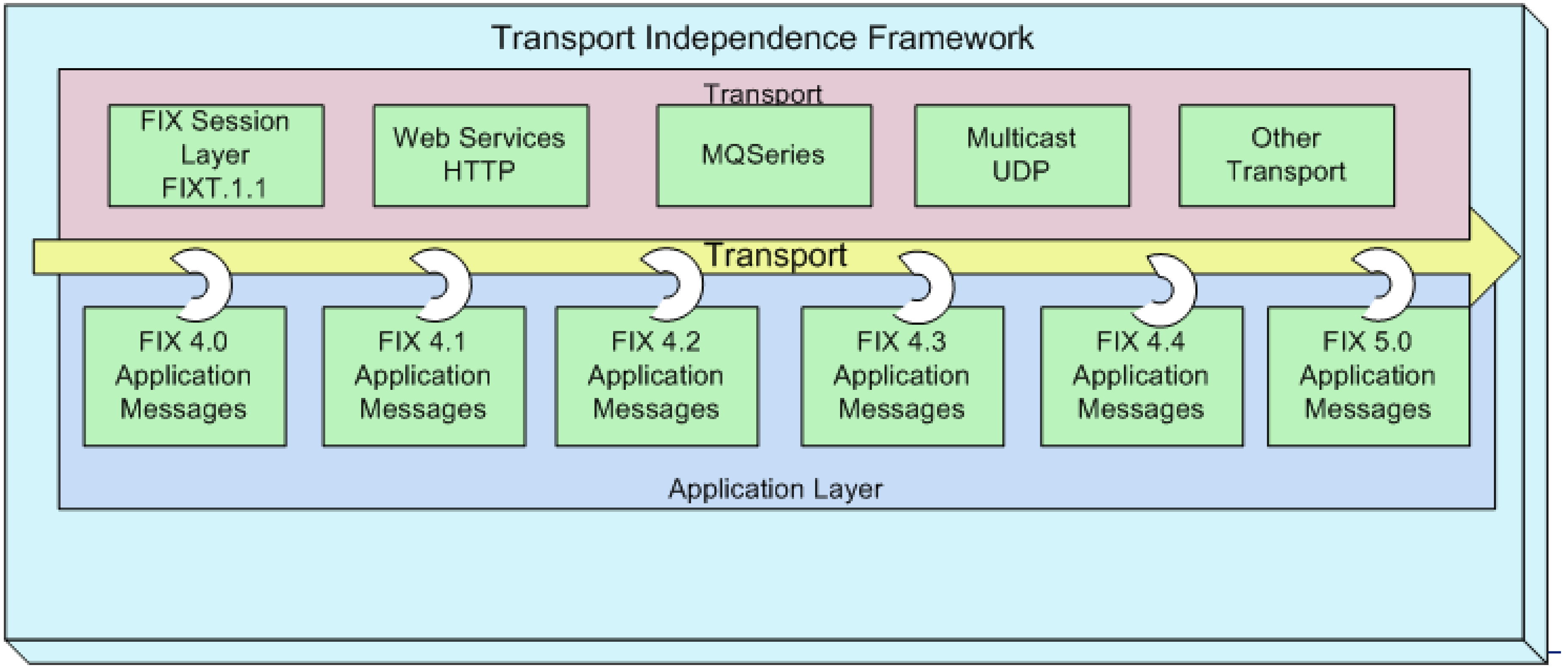
- The release of FIX 5.0 introduced the Transport Independence (TI) framework that separates the FIX Session Protocol from the FIX Application Protocol
- Under the TI framework, application messages can be sent over any suitable session transport technology (e.g. WS-RX, MQ, message bus)
- FIX Session and Application layers have been decoupled and are now independent. Each has its own versioning moniker (i.e. FIXT.1.1, FIX.5.0)
- TI treats the FIX Session like ‘any other’ transport
- The FIX Session Protocol is one of the available options as a session transport for application messages

FIX 5.0 Transport Independence



Transport Independence (TI) Framework

- Separates the FIX Session layer from the FIX Application layer
- The FIX Application Protocol can use any transport technology in addition to the FIX Session Protocol


Transport Independence Framework
Transport
FIX Session Layer FIXT.1.1
Web Services HTTP
MQSeries
Multicast UDP
Other Transport
Transport
FIX 4.0 Application Messages
FIX 4.1 Application Messages
FIX 4.2 Application Messages
FIX 4.3 Application Messages
FIX 4.4 Application Messages
FIX 5.0 Application Messages
Application Layer

NYSE Technologies

New FIX Session Protocol Version 1.1

FIXT 1.1 is one of several supported Transport layers for exchanging FIX application messages. It consists of very minor changes to the FIX Session Protocol for FIX 4.0 - 4.4

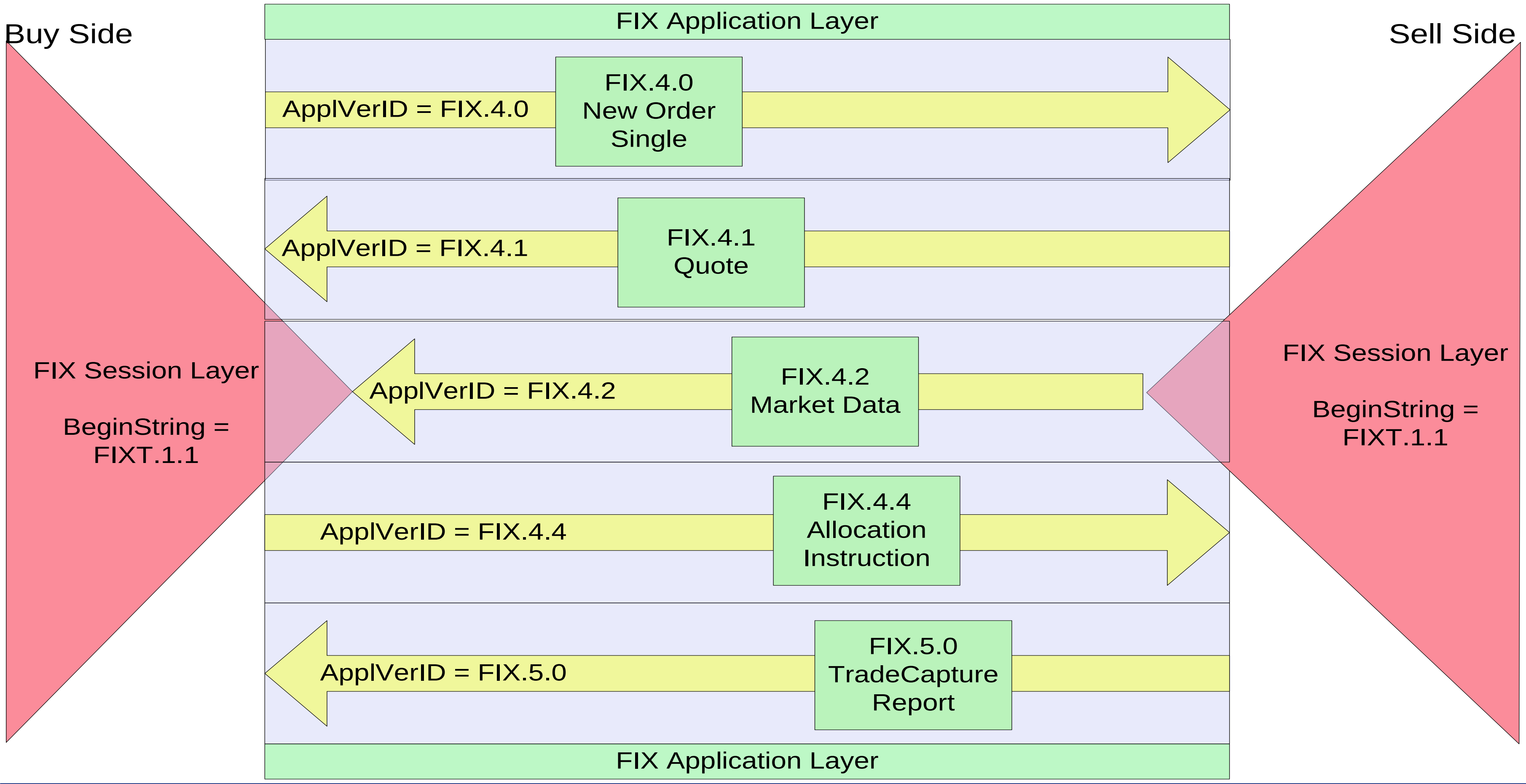
Additions to the **Standard Message Header:**

- ***ApplVerID(1128)*** – indicates the FIX application version using a service pack identifier. *ApplVerID* applies to each message individually
- ***CstmApplVerID(1129)*** – the custom protocol extension to which this message belongs. Used to support bilaterally agreed custom functionality

Additions to the **Logon message**

- ***DefaultApplVerID(1137)*** – the default version of FIX carried over this FIXT session

Flexibility Provided by FIX 5.0



FIX 5.0 Extensions

- Comprehensive Post-Trade Support
- Vast Order Routing Toolbox
- Algorithmic Trading Support Kit (FIXatdl)
- Market Data for Efficient Book Management
- Expanded FX Quoting, Trading, and Streaming Data
- MiFID, RegNMS, OATS Compliance

FIX 5.0 Technology Framework Components

The Technology Framework consists of

- FIX Session Layer
- FIX Syntax
- Message Repository
- FIXML Schema
- FAST Compression
- FAST Session Layer
- Transport Independence (FIX 5.0)
- Application Versioning (FIX 5.0)
- FIXimate (improved in FIX 5.0)

FIX Adapted for Streaming (FAST Protocol)

- The increased volume for Market Data over FIX
- 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

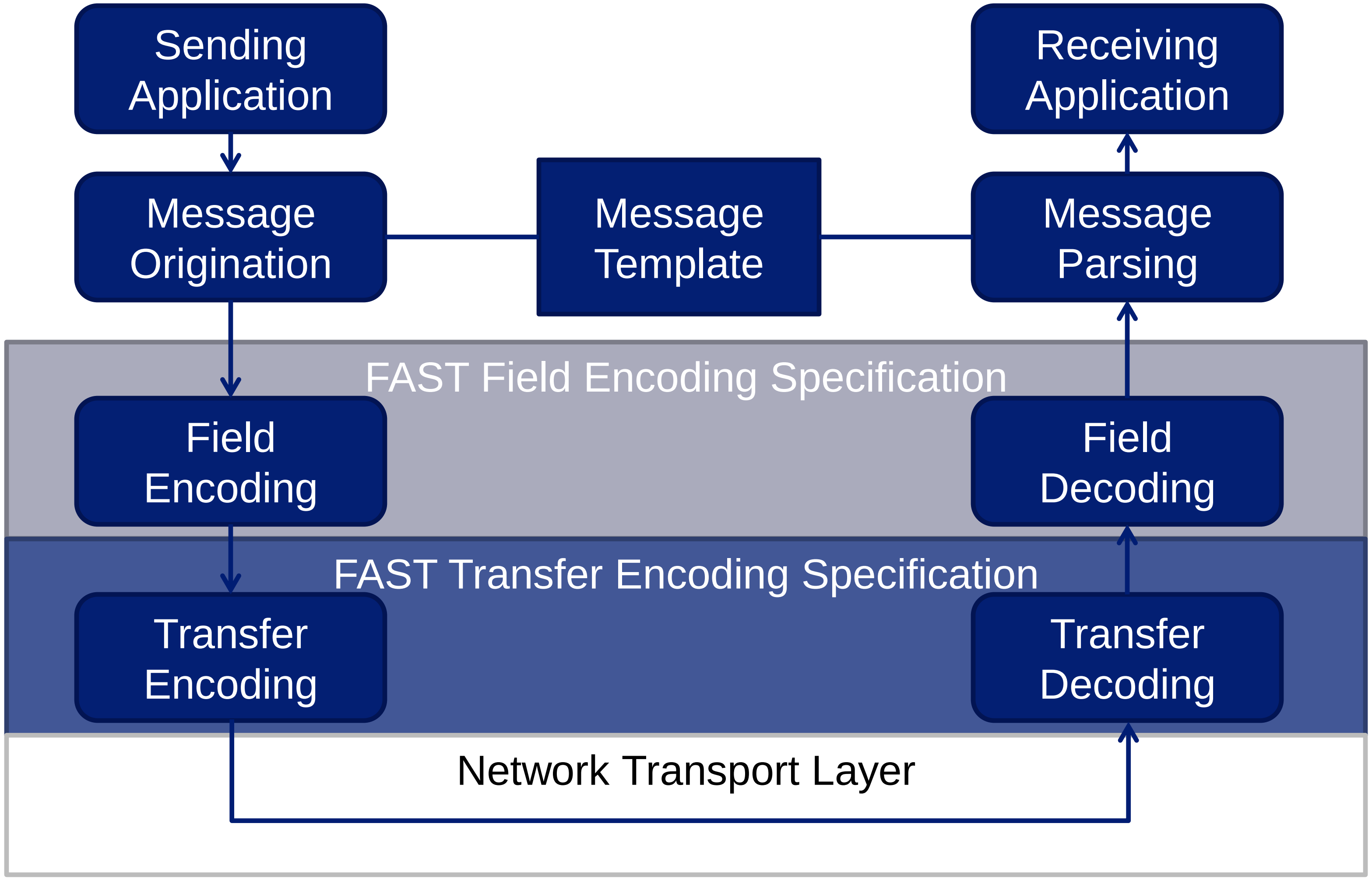
FIX Adapted for Streaming (FAST Protocol)

- The FIX Adapted for STreaming (FAST) Protocol is designed to use as little bandwidth and minimize processing latency
- The solution comprises of certain algorithms to compress the message using a pre-agreed template prior to sending it
- The receiving engine decompresses the message using the same template before forwarding it to the trading application.
- This results in less bandwidth and latency on the network
- However, there is some impact on the CPU as it compresses and decompresses the message

FAST Protocol – Templates

- Compression / decompression method is determined by use of a template
- Same template is exchanged out-of-band and used by both counterparties
- Indicates how a message should be compressed

FAST Protocol Layers



CME FAST – Efficiencies in Practice

Peak requirements today

- Bandwidth = 10 Mbps
- Message Rate = 10k MPS

Projections for Q4 2007 (peak)

- Bandwidth = 25 Mbps
- Message Rate = 30k MPS

Projections with FIX and FAST

- 70% bandwidth reduction
- 50% less data content
- 60% fewer messages

Cost savings is expected to be substantial

Other FIX Concepts – FIXML

What is FIXML?

Two ways of using FIXML:

- FIXML over FIX
- FIXML-to-FIX Translation

What is FIXML?

FIXML is the XML vocabulary for creating FIX messages

- Uses the same FIX data dictionary and business logic
- Focuses on the FIX Application messages – does not provide a Session layer

Instead of sending an entire FIX message, it requires

- Header Tags – BeginString(8), BodyLength(9), MsgType(35), MsgSeqNum(34), XmlDataLen(212), and XmlData(213)
- Trailer Tags – CheckSum(10)

First working version (1.1) in 1/15/1999 based on FIX 4.1

- Moved from DTD (Document Type Definition) to XML Schema in 2004 to reduce bandwidth (FIXML 4.4)

What is FIXML?

- FIXML or XML content is enclosed in the standard Header via XmlDataLen(212) and XmlData(213) fields followed the standard Trailer
- The generic MsgType value of ‘n’ for "XML message can be used when transmitting XML content that is not defined with a FIX message type
- Requires content to be ordered – different than traditional FIX “<tag>=<value>” syntax, which allows fields to be in any order
- Validation of these attributes must happen at the application level

FIXML Over FIX

Sample FIXML Order Message:

- <FIXML xmlns="http://www.fixprotocol.org/FIXML-4-4" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fixprotocol.org/FIXML-4-4/../../schema/fixml-main-4-4.xsd" v="4.4" r="20030618" s="20040109">
- <Order ID="123456" Side="1" TxnTm="2003-12-18T12:00:00" Typ="2" Px="85.00">
- <Hdr TID="SSB" SID="FCM" SeqNum="1" Snt="2003-12-18T12:00:00"/>
- <Instrmt Sym="IBM"/>
- <OrdQty Qty="100"/>
- </Order>
- </FIXML>

Why FIXML ?

- XML is becoming the meta-language for standards
- Other in-house applications may use XML, encouraging software reuse
- Transporting FIXML internally may make sense as well as using it externally
- You may wish to use a middleware or other protocols instead of the FIX session transport, e.g. JMS, IBM MQSeries, TIBCO Rendezvous
- You may want to use FIX with other XML protocols (FpML, OFX, etc.)
- Trading or vendor technology partners may want to use it

Why Not FIXML ?

- Bandwidth overhead: FIXML requires much more bandwidth to transmit the same amount of data. FIXML may take more than 5 times the bandwidth of FIX
- Performance overhead: XML parsers, especially validating parsers, might be slower than simple FIX engines
- Support: FIXML is not presently widely used and supported

What is FIXatdl?

- Algorithmic Trading Definition Language
- Simplifies integration of strategies with a standard way of describing algorithm strategies
- Allows OMS/EMS vendors and users to deploy trading algorithms without re-coding, re-testing, and certification
- Uses XML and FIX to define parameters of algorithmic order types, dependencies, and rules for validating the values entered into an order by a user

Why was FIXatdl introduced?

- Implementation of Algorithms in FIX is static
  - FIX defines ~2 dozen Order Types (tag 40) and Execution Instructions (tag 18) which OMS/EMS vendors implemented
  - Algorithms in last few years were implemented in the same way using UDFs to call undefined behavior
- New algorithms require coding, testing and certification
- OMS / EMS users needed to wait for new releases and upgrade to leverage new strategies

How does FIXatdl function?

- Allows you to describe the FIX tags and data necessary to support trading of algorithmic order types, including
  - Ability to assign a FIX tag number to a parameter
  - Ability to declare the data type of a parameter
  - Ability to constrain the values entered by the user (i.e. valid values, conditional rules)
  - Ability to validate parameter values entered by the user via simple bounds checking (i.e. via application or engine)
  - Ability to initialize the value of GUI control object

How does FIXatdl work?

- Brokers describe the inputs to their algorithms in an XML document
- Applications receive XML documents and dynamically display an order entry ticket containing the algorithmic order parameters in a GUI
- OMS allow users to input order values on the fly
- Data entered by the user is captured in the appropriate FIX tags described in the broker's XML document and sent to the broker

NYSE Technologies℠

Powering the exchanging world.℠

Module XI: Case Studies

Case Study – Buy-side Firm

Problem Statement: Buy-side A wants to automate its electronic trading process

Several items must be considered:

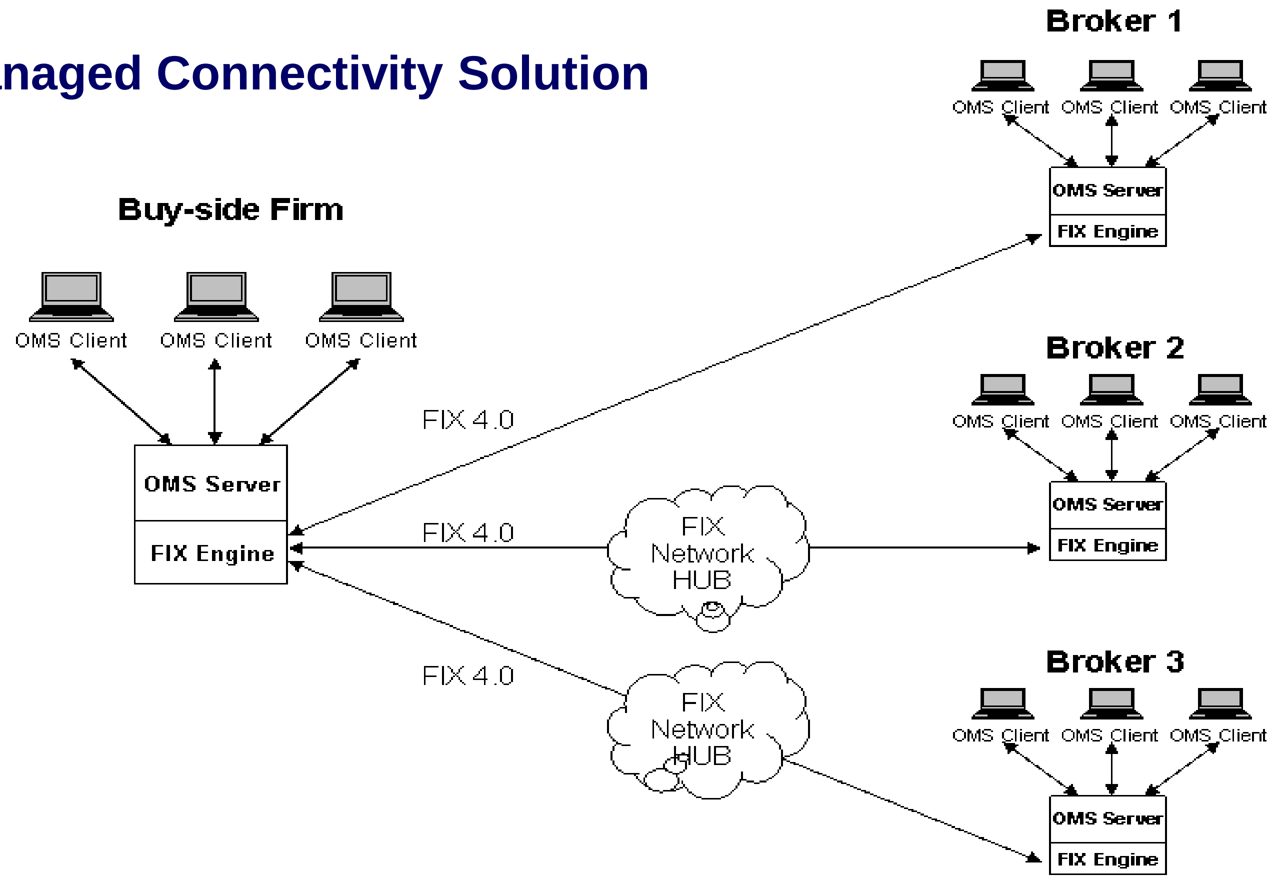
- What business flow do they want to automate (front office, asset classes, back office)?
- Select a network: routing network; leased line; Internet?
- Select a FIX engine or FIX-enabled trading application?
- Outsource or self-manage? (may depend on resource availability)

Solutions:

- Self-managed connectivity to brokers (point-to-point and hosted connections)
- Outsourced connectivity to brokers

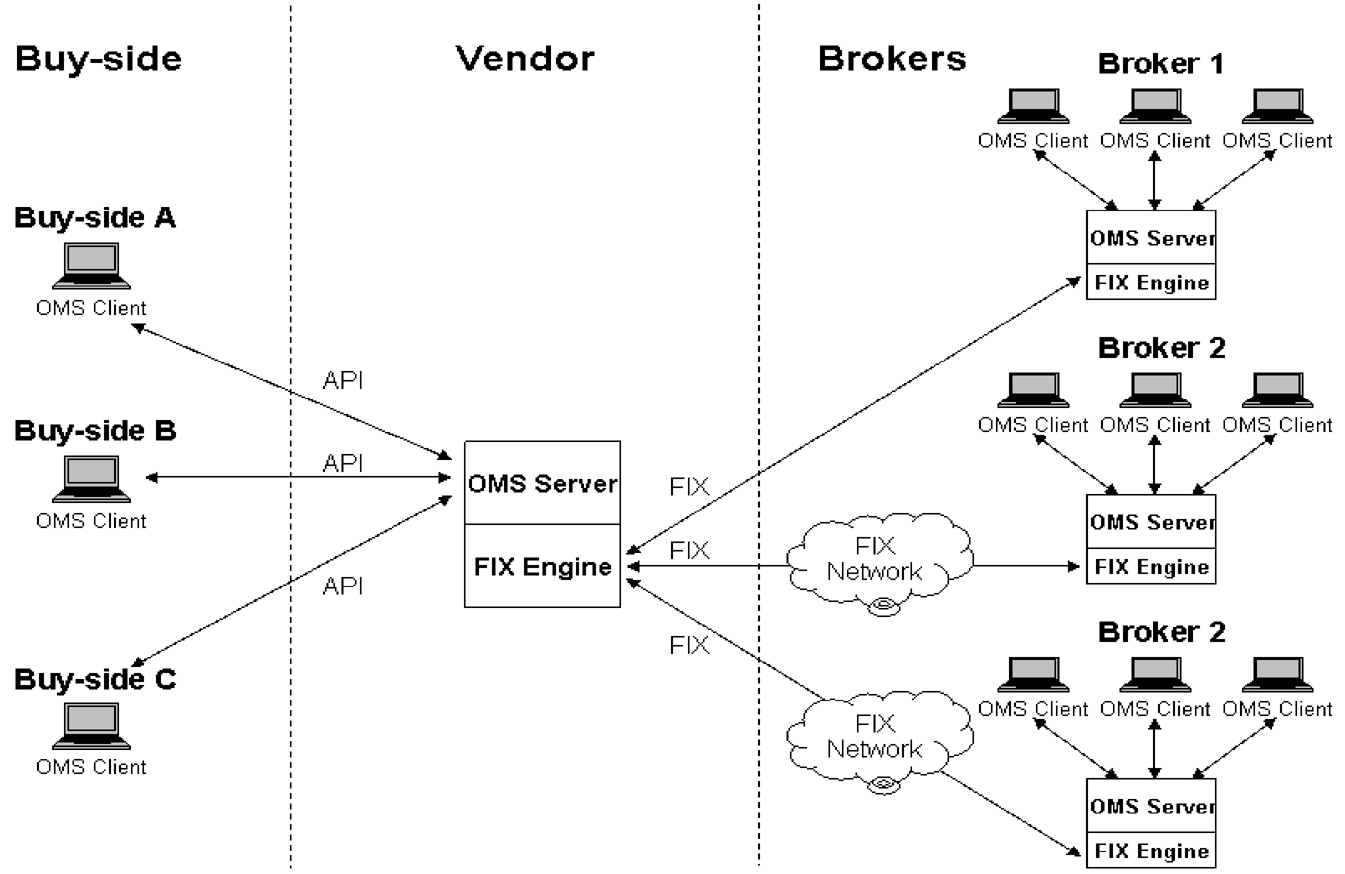
Case Study – Buy-side Firm

Self-managed Connectivity Solution



Case Study – Buy-side Firm

Outsourced Connectivity Solution



NYSE Technologies™

Case Study – Sell-side Firm

Problem Statement: Sell-side A wants to automate its electronic trading process and receive client orders electronically

Several items must be considered:

- Must accommodate various client FIX versions and nuances
- Select a network: routing network; leased line; Internet?
- Select a FIX engine or FIX-enabled trading application?
- Outsource or self-manage? (may depend on resource availability)

Case Study – Sell-side Firm

Prior to using FIX

- FTP
- Telephone
- SWIFT
- Fax

Types of messages

- Equity trading (Single Orders, List Orders, Cancels, Executions)
- Indications of Interest
- Futures

Case Study – Sell-side Firm

Many reasons for using FIX:

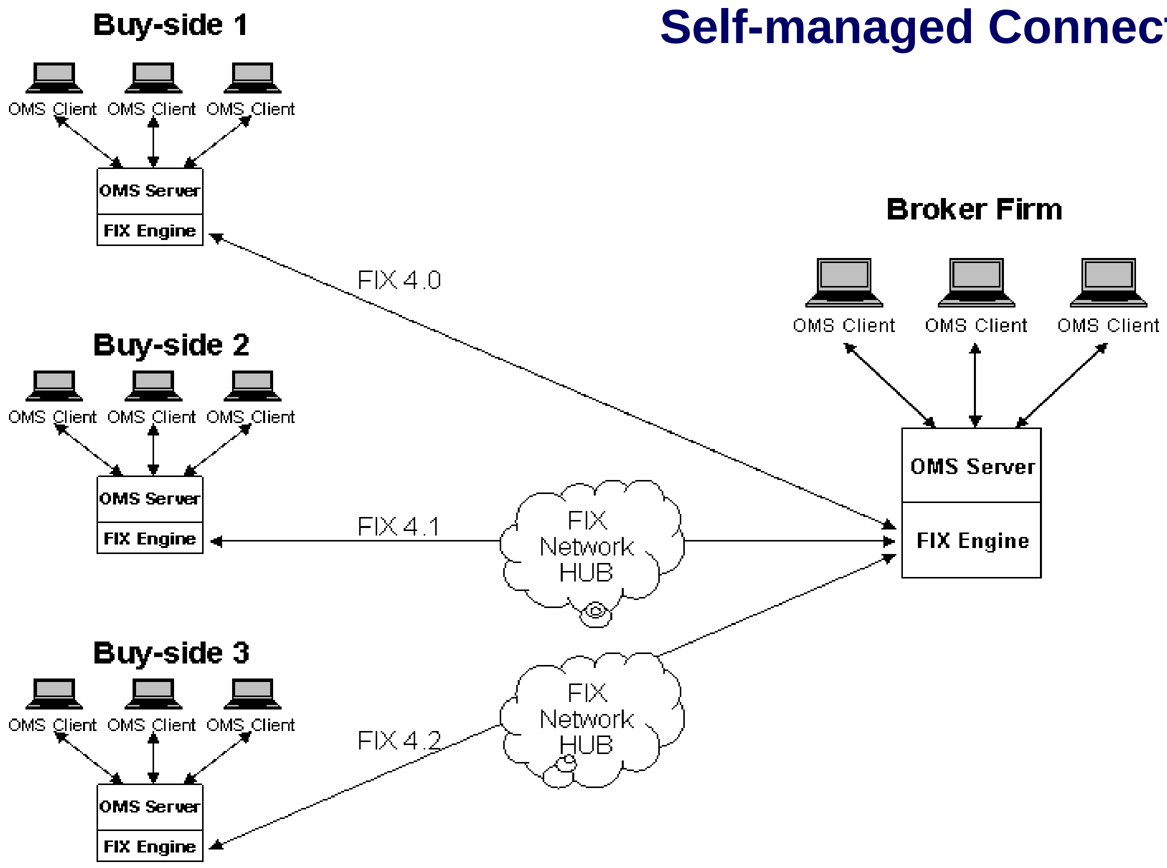
- Client demand
- Ease of on-boarding
- Industry standard
- Access to counterparties worldwide
- One technology for all counterparty connections

Solutions:

- Self-managed connectivity to buy-sides and execution venues
- Outsourced connectivity to buy-sides and execution venues

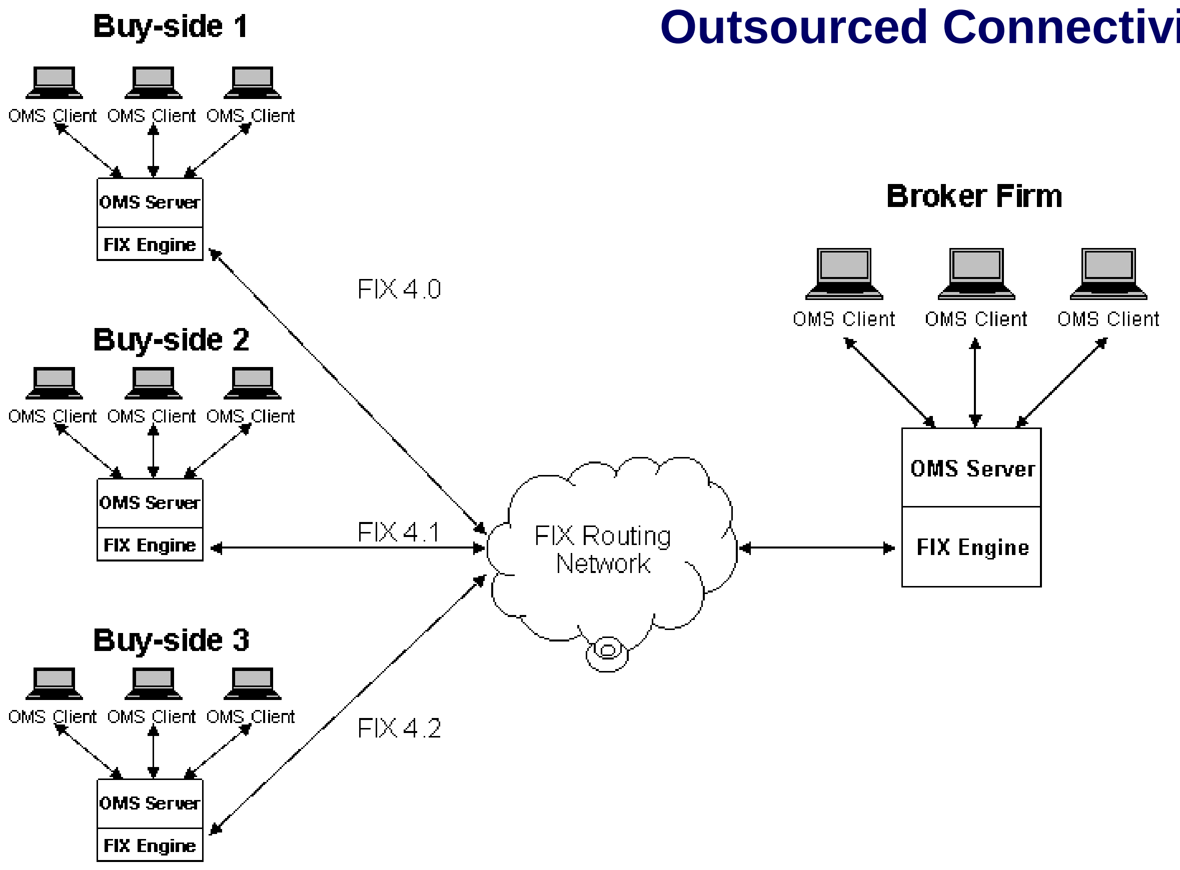
Case Study – Sell-side Firm

Self-managed Connectivity Solution



Case Study – Sell-side Firm

Outsourced Connectivity Solution



Case Study – Network / Global Broker

Problem Statement: Large, multi-national broker wants to automate its global electronic trading process and receive client orders electronically

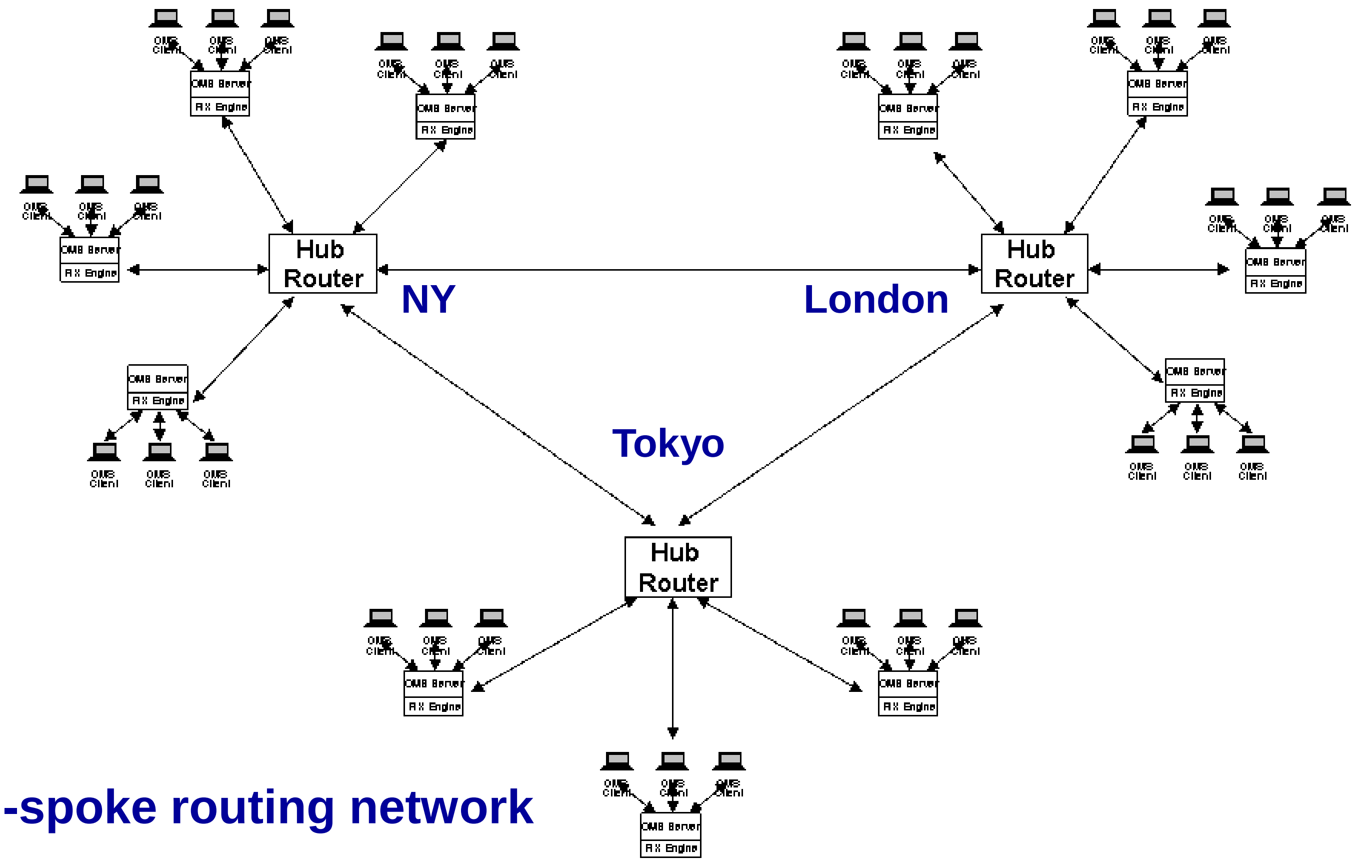
Several items must be considered:

- Global versus local routing
- Must accommodate various client FIX versions and nuances
  - FIX version translation
  - Message normalization
  - Symbology conventions
- Multiple trading applications (multi-asset, multiple desks, etc.) to connect

Solution:

Hub-and-spoke routing network

Case Study – Network / Global Broker



Hub-and-spoke routing network

NYSE Euronext

Powering the exchanging world.SM