

# FIX Protocol Primer

**FIX**PROTOCOL  
INDUSTRY-DRIVEN MESSAGING STANDARD™



# FIX Protocol Primer

**FIX**PROTOCOL  
INDUSTRY-DRIVEN MESSAGING STANDARD™

**Scott Atwell**

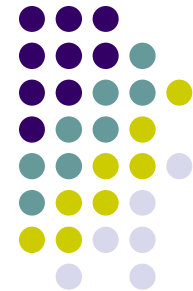
Co-Chair FIX Protocol Global Steering Committee  
American Century Investments

**Jim Northey**

Co-chair FPL Americas Regional Committee  
The LaSalle Technology Group

**Ryan Pierce**

FIX Technical Director  
FIX Protocol Ltd.





## Agenda

- FIX Overview
- FIX Syntax
- FIX Session Layer
- FIX Semantics

# FIX Overview

**FIX**PROTOCOL  
INDUSTRY-DRIVEN MESSAGING STANDARD™





## FIX Features

- FIX is the pre-eminent global messaging protocol for the pre-trade and trade environment up to pre-settlement for securities
- FIX is an open standard for multiple security classes, facilitating greater STP
- FIX provides seamless connectivity to multiple counter-parties and enhances price transparency and access to liquidity. It is a superior solution that reduces reliance on broker and vendor proprietary solutions
- It enables accurate real time updating of portfolio and trading management systems for pre-trade, trade and pre-settlement
- The FIX Protocol is owned and maintained by FIX Protocol Limited, a not-for-profit industry association of approximately 200 firms from across the financial services sector. Representatives from these firms dedicate time and resource to ensure that the FIX Protocol is continually enhanced to meet the ever evolving needs of this sector



## FIX Protocol Highlights

- The FIX Protocol is an open protocol specification (not a software implementation) developed via an open process by the industry participants who use FIX for their business
- The FIX Protocol celebrated its 15 year anniversary this year (2009)
- FIX Protocol versions:
  - 2.7, 3.0, 4.0, 4.1, 4.2, 4.3, 4.4,
  - 5.0, 5.0SP1, 5.0SP2
  - FIXT1.1
- Geographic/product expansion (in order): U.S., Europe, Japan, Asia/Pac, Derivatives, Fixed Income, Foreign Exchange
- There are 250 vendors with FIX-based products and services listed on the FIX website. Open source FIX software also exists.



## FIX as a solution has expanded considerably

- FIX started within the buy-side to sell-side to communicate indication of interest (IOIs) then order routing for equities
- Quickly (a matter of a 2-3 years) added additional asset classes
  - FX
  - Listed Derivatives
  - Fixed Income
- Quickly added additional business areas and processes
  - Allocations
  - Quote driven market support
  - Market data
  - Exchange / ATS / ECN support
  - Reference data
  - Trade capture and trade reporting
  - Listed derivatives clearing
- Pre-trade through post-trade pre-settlement across multiple asset classes



# FIX Support for Equities and Derivatives

Message Support	FIX Version							
	4.0	4.1	4.2	4.3	4.4	5.0	5.0 SP1	5.0 SP2
<b>Equities</b>								
Basic Order Flow	Good	Good	Good	Good	Good	Good	Good	Good
IOI's and Advertisements	Good	Good	Good	Good	Good	Good	Good	Good
Quotes	Good	Good	Good	Good	Good	Good	Good	Good
Market Data	No	No	Good	Good	Good	Good	Good	Good
Allocations	Some	Some	Good	Good	Good	Good	Good	Good
Confirmations/Affirmations	No	No	No	No	Good	Good	Good	Good
Trade Reporting	No	No	No	Some	Good	Good	Good	Good
Program Trading	Some	Some	Good	Good	Good	Good	Good	Good
Algorithmic Trading	Some	Some	Some	Some	Good	Good	Good	Good
<b>Futures and Options</b>								
Basic Order Flow	No	Some	Good	Good	Good	Good	Good	Good
Multi-leg Order Flow	No	No	Some	Good	Good	Good	Good	Good
IOI's and Advertisements	Some	Good	Good	Good	Good	Good	Good	Good
Quotes	No	Some	Good	Good	Good	Good	Good	Good
Market Data	No	No	Good	Good	Good	Good	Good	Good
Allocations	No	Some	Good	Good	Good	Good	Good	Good
Confirmations/Affirmations	No	No	No	No	Good	Good	Good	Good
Trade Reporting	No	No	No	Some	Good	Good	Good	Good
Security & Position Reporting	No	No	No	No	Good	Good	Good	Good
Collateral Management (Listed Derivatives)	No	No	No	No	Good	Good	Good	Good

<b>Legend</b>	No Support	Some Support	Good Support	Not Applicable
---------------	------------	--------------	--------------	----------------

NOTE: levels of support are based on fields incorporated into the core protocol at a given stage. This rating does not include custom fields, which can be added to earlier versions of the protocol to achieve given functionality.



# FIX Support for Fixed Income and Foreign Exchange

Message Support	FIX Version							
	4.0	4.1	4.2	4.3	4.4	5.0	5.0 SP1	5.0 SP2
<b>Fixed Income</b>								
Basic Order Flow	Red	Red	Yellow	Green	Green	Green	Green	Green
Multi-Leg Order Flow (Repos, Swaps, Switches, Rolls)	Red	Red	Red	Yellow	Green	Green	Green	Green
IOI's (Offerings)	Yellow	Yellow	Yellow	Green	Green	Green	Green	Green
Quotes	Red	Yellow	Yellow	Green	Green	Green	Green	Green
Allocations	Red	Yellow	Yellow	Green	Green	Green	Green	Green
Confirmations/Affirmations	Red	Red	Red	Red	Green	Green	Green	Green
Trade Reporting	Red	Red	Red	Yellow	Green	Green	Green	Green
Collateral Management	Red	Red	Red	Red	Yellow	Yellow	Yellow	Yellow
<b>Foreign Exchange</b>								
Basic Order Flow (Spots, Forwards)	Red	Yellow	Green	Green	Green	Green	Green	Green
Basic Order Flow (Swaps)	Red	Yellow	Yellow	Yellow	Yellow	Green	Green	Green
Quotes (spots, outright forwards, FX swaps)	Red	Yellow	Green	Green	Green	Green	Green	Green
Market Data (executable streaming prices)	Red	Yellow	Yellow	Yellow	Yellow	Green	Green	Green
Allocations	Red	Yellow	Yellow	Yellow	Yellow	Yellow	Yellow	Yellow
Confirmations/Affirmations	Red	Red	Red	Red	Yellow	Yellow	Yellow	Yellow
Trade Reporting	Red	Red	Red	Yellow	Yellow	Yellow	Yellow	Yellow

<b>Legend</b>		No Support		Some Support		Good Support		Not Applicable
---------------	---	------------	---	--------------	--	--------------	---	----------------

NOTE: levels of support are based on fields incorporated into the core protocol at a given stage. This rating does not include custom fields, which can be added to earlier versions of the protocol to achieve given functionality.



# FIX Markets and General Support

Message Support	FIX Version							
	4.0	4.1	4.2	4.3	4.4	5.0	5.0 SP1	5.0 SP2
<b>Exchanges and Markets</b>								
Market Data (Including Price Dissemination)	Red	Red	Green	Green	Green	Green	Green	Green
Reference Data (Instruments and Products)	Red	Red	Yellow	Yellow	Green	Green	Green	Green
Parties Reference	Red	Red	Red	Red	Red	Red	Red	Green
Market Structure	Red	Red	Red	Red	Red	Red	Green	Green
Quote Driven Markets	Red	Red	Green	Green	Green	Green	Green	Green
Order Driven Markets	Green	Green	Green	Green	Green	Green	Green	Green
OTC derivatives trading and clearing	Red	Red	Red	Red	Red	Red	Yellow	Green
<b>General</b>								
News	Green	Green	Green	Green	Green	Green	Green	Green
Email	Green	Green	Green	Green	Green	Green	Green	Green
Transport Independence Framework	Blue	Blue	Blue	Blue	Blue	Green	Green	Green
Regulatory Compliance (MiFID, Reg NMS, Reg SHO, OATS, ACT, NYSE 80A)	Blue	Blue	Blue	Blue	Blue	Green	Green	Green

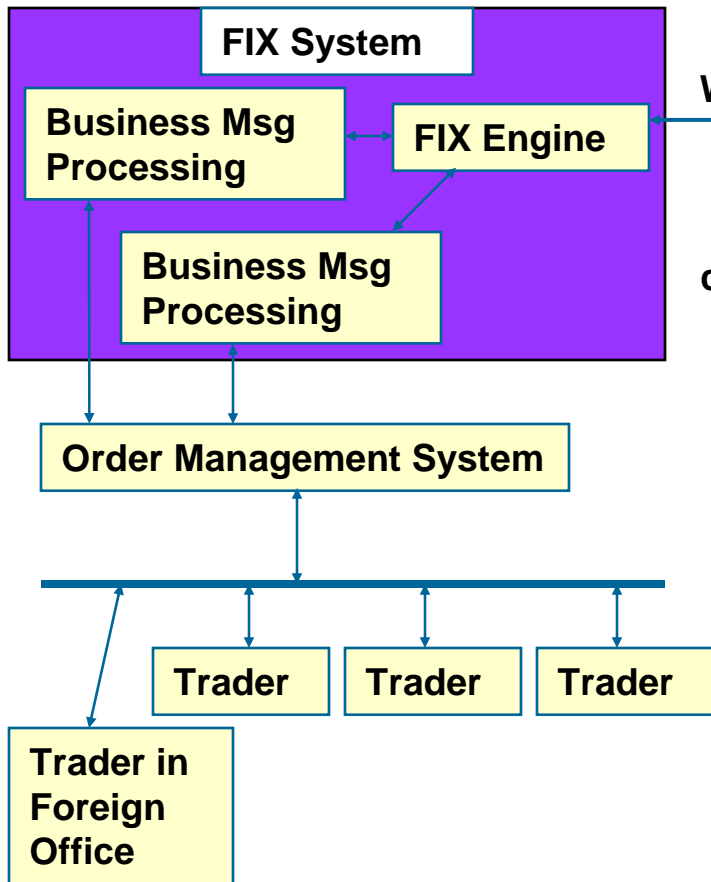
<b>Legend</b>	<span style="display:inline-block; width:15px; height:15px; background-color:red; border:1px solid black;"></span> No Support	<span style="display:inline-block; width:15px; height:15px; background-color:yellow; border:1px solid black;"></span> Some Support	<span style="display:inline-block; width:15px; height:15px; background-color:green; border:1px solid black;"></span> Good Support	<span style="display:inline-block; width:15px; height:15px; background-color:blue; border:1px solid black;"></span> Not Applicable
---------------	---	--	---	--

NOTE: levels of support are based on fields incorporated into the core protocol at a given stage. This rating does not include custom fields, which can be added to earlier versions of the protocol to achieve given functionality.

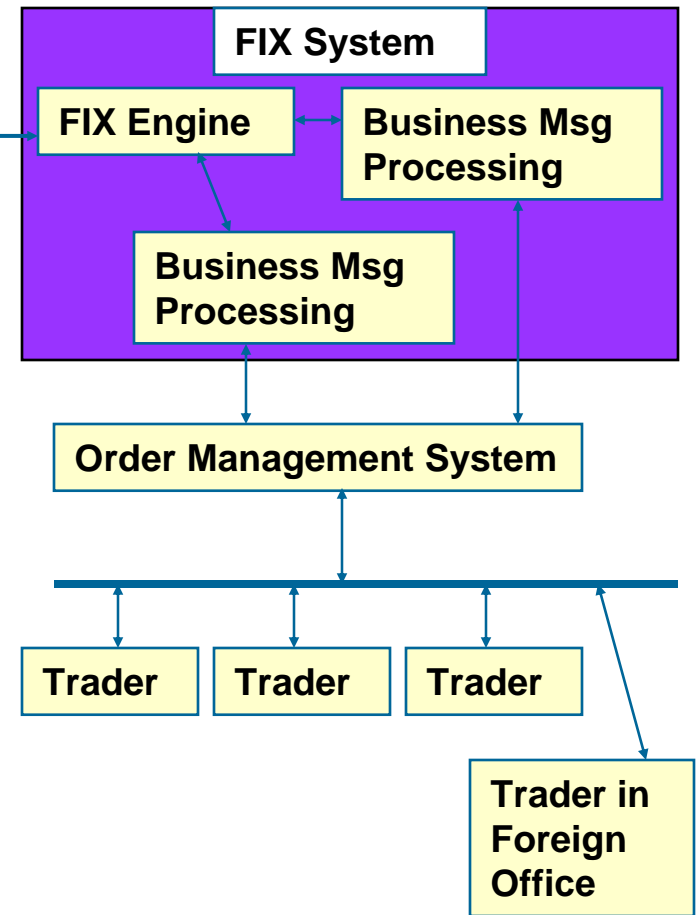


# FIX System Connectivity

## Customer (i.e. Investment Mgr)



## Supplier (i.e. Broker/Dealer)



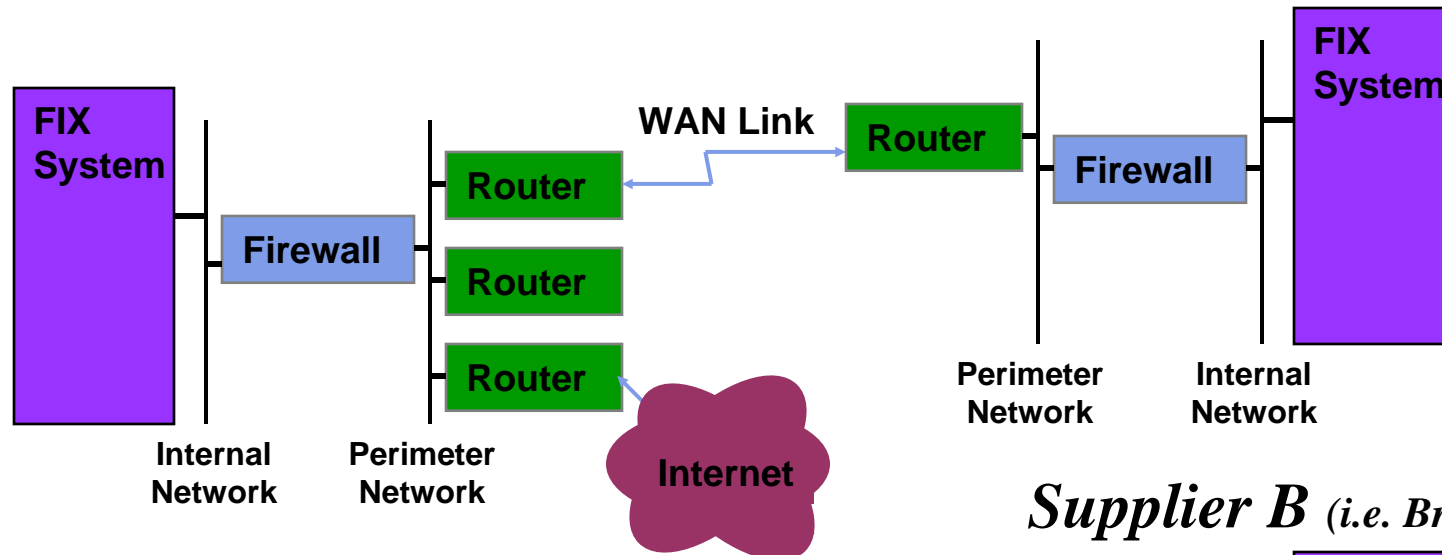
Wide Area Network Link  
TCP/IP  
(TCP Socket opened by customer, persists during life of FIX session)



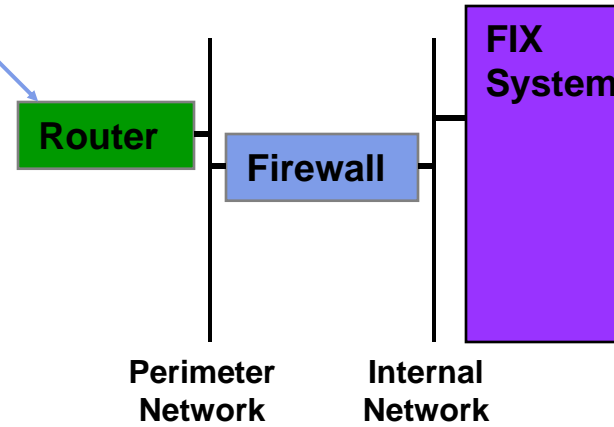
# Typical FIX Connectivity – Point to Point

*Customer (i.e. Investment Mgr)*

*Supplier A (i.e. Broker/Dealer)*



*Supplier B (i.e. Broker/Dealer)*



**Note: “WAN Link” could be:**

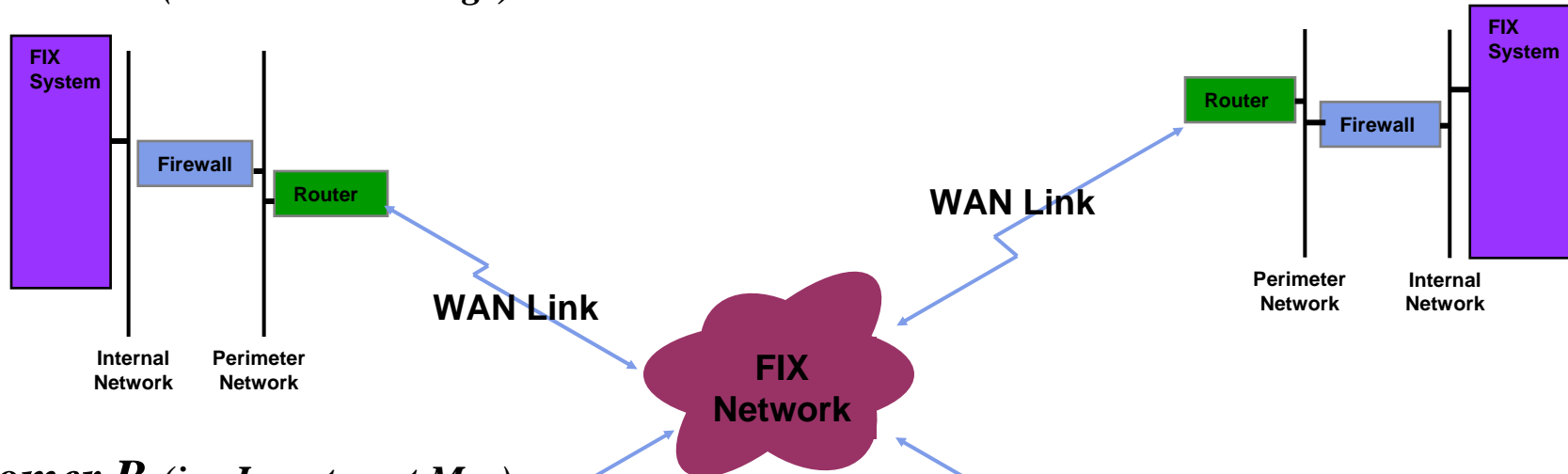
- A dedicated circuit (i.e. 56KB leased line, Frame Relay, etc.)
- A shared network (i.e. network provider, Virtual Private Network, etc.)
- The Internet



# Typical FIX Connectivity – Shared Network

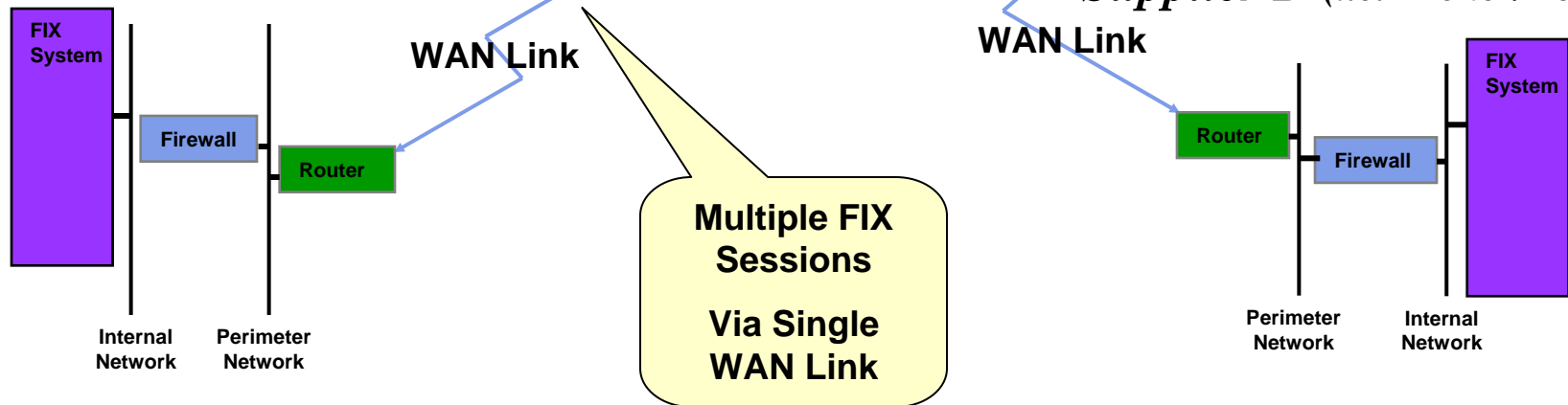
*Customer A (i.e. Investment Mgr)*

*Supplier A (i.e. Broker/Dealer)*



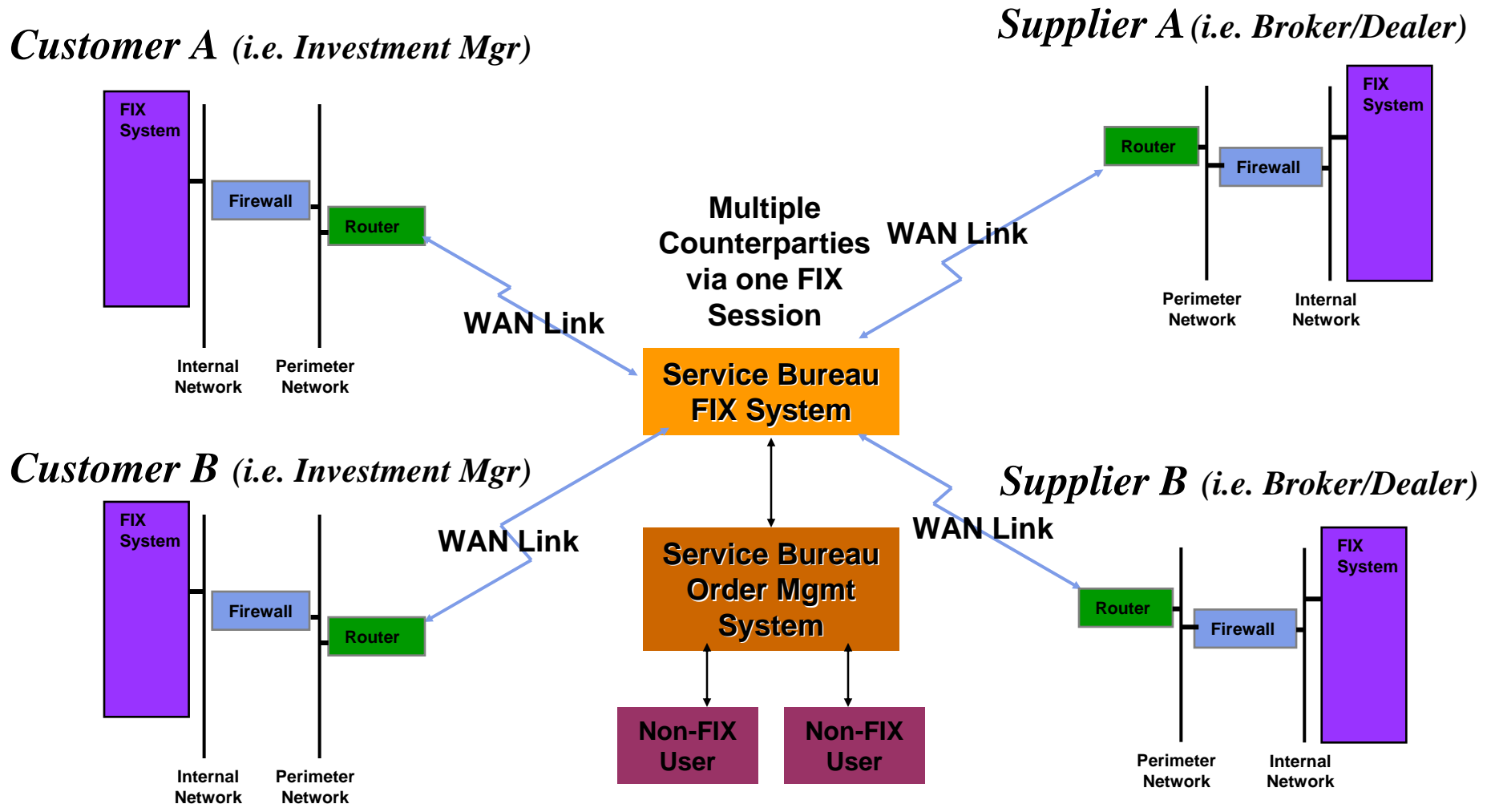
*Customer B (i.e. Investment Mgr)*

*Supplier B (i.e. Broker/Dealer)*





# Typical FIX Connectivity – Service Bureau





## FIX via Service Bureau

- Advantages
  - Fewer FIX connections and sessions to manage
  - Service bureau can pre-test counterparties
  - Opportunity to convert from one version/format to another
- Disadvantages
  - More eggs in one basket
  - Service bureau “knows” your trading data
  - Difficult to know if and which counterparties are connected to service bureau
  - Can affect performance

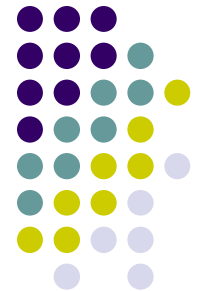


## FIX Connectivity - Key Points

- Use of private network, shared network (VPN), or Internet look identical to a FIX engine
- Similar network infrastructure and design
  - i.e. Same desire and need for firewalls
- Similar security (authentication/encryption) concerns
  - Unwise to send highly sensitive data unencrypted via private network or shared network

# FIX Syntax

**FIX**PROTOCOL  
INDUSTRY-DRIVEN MESSAGING STANDARD™





## Introduction to the FIX Syntax - Agenda

- FIX Building Blocks
  - Fields
    - Enumerations
  - Messages
  - Components
- FIX Tag=Value Syntax
  - Format
  - Field order
  - Repeating groups
  - Data fields
- FIXML Syntax
  - Design Rules
  - Examples



## FIX Building Blocks

- Fields
  - Data items that usually contain a single value
  - Identified by a FIX Tag Number
    - 1 to 5000 – Standard FIX Tag Numbers
    - 5000-9999 – Custom Fields – Publicly Identified
    - 10000+ - Private user defined fields
  - Each field has a *FIX Datatype*
  - ASCII data only i.e., Cannot contain the `<SOH>` character `0x01`
    - Except for certain fields of the FIX datatype `data`
- Components
  - Groups of Fields
- Messages
  - Fields
  - Components



## Tag = Value

- Versions
  - FIX4.0
    - **DO NOT USE**
  - FIX.4.1
    - Still in use
  - FIX.4.2
    - Widely used
  - FIX.4.3
    - **DO NOT USE**
  - FIX4.4
    - Widely used
  - FIX5.0 (SP2 just released)
    - Initial adoption underway by exchanges
  - FIXT1.1
    - Session layer only



## The FIX Tag = Value Syntax

- Composed of four parts
  - *Tag*
    - FIX Field Tag Number
  - **=**
  - *Value*
  - Delimiter
    - non-printable ASCII character known as Start of Header (SOH)
    - character *0x01* displays on the computer as *^A*
    - Often displayed in documentation using one of the following :

*TAG=VALUE^A*

|  
^  
<SOH>

- Example of the first field in a message

8=FIX.4.1<SOH>

- BeginString – identifies the version of the FIX message

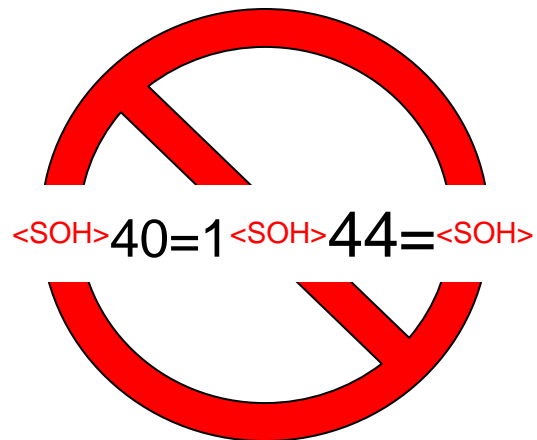
<SOH>44=100.5<SOH>

- Price – identifies the price



## No Empty Fields

- Empty fields are not permitted
- All fields must have a value
- Optional fields without values are not included
- Messages with empty fields should be rejected
  - Send a Session Level Reject in response





## Tag = Value Field Order Rules

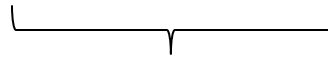
- A Field (tag) can only appear in a message one time EXCEPT:
  - Within a repeating group (next slide)
- Fields within a FIX message can appear in any order EXCEPT:
  - Start of Message  
**8=FIX.4.2<SOH>9=182<SOH>35=D<SOH>...**
    - The first three fields in the standard header
      - BeginString (tag 8)
      - BodyLength (tag 9)
      - MsgType (tag 35)
  - End of Message  
**.... <SOH>10=xxx<SOH>**
    - The last field in the standard trailer is the CheckSum (tag 10)
    - Notice the trailing **<SOH>** character
  - Repeating groups
    - The order of fields within repeating groups must be preserved



## Data Content

- ASCII Characters
  - Field must not contain **<SOH>** Character 0x01

**<SOH>**58=12345**<SOH>**7890**<SOH>**



Data with embedded **<SOH>**  
characters in the Text (tag 58) field







## Repeating Groups

- Repeating groups permit a set of FIX fields to be repeated within a message
  - Notable uses:
    - Parties block
    - Alternative security identifiers
    - Legs of a multileg instrument
    - Individual allocations for an order
- Example – The Parties Block

Tag or Component	Field Name	FIXML name
<i>Repeating Group 453</i>	<b>NoPartyIDs</b>	
448	PartyID	@ID
447	PartyIDSource	@Src
452	PartyRole	@R
Component	<b>PtysSubGrp</b>	Sub
<i>end Repeating Group</i>		



## Repeating Groups Rules

- Start with a NumInGroup field NoXxxxx
  - where “No” is the abbreviation for “Number”
  - Examples:
    - NoPartyIDs(tag 453)
    - NoMiscFees(tag 136)
    - NoLegs(tag 555)
- The first field listed after the NoXxxx field **MUST BE** provided
  - Acts as the delimiter for each repeating group
  - Example:
    - PartyID(tag 448) must occur after NoPartyIDs(tag 453)
- Do not include the NoXxxx field if there are zero entries
  - NoXxxx=0 is permitted – NOT recommended
- Fields within a repeating group must occur in the order specified in the FIX Specification



## Repeating Group Example

Tag or Component	Field Name	FIXML name
Repeating Group 453	<b>NoPartyIDs</b>	
448	PartyID	@ID
447	PartyIDSource	@Src
452	PartyRole	@R
Component	<b>PtysSubGrp</b>	Sub
end Repeating Group		

- Identify the executing broker and the clearing firm on a FIX.4.4 order

NoPartyIDs(tag 453)=2

PartyID(tag 448)=DEUTDEFF500

PartyIDSource(tag 447)=B *note: Bank Identifier Code (BIC) ISO 9362*

PartyRole(tag 452)=1 *note: Executing Firm*

PartyID(tag 448)=GSILGB2XHUL

PartyIDSource(tag 447)=B *note: Bank Identifier Code (BIC) ISO 9362*

PartyRole(tag 452)=4 *note: Clearing Firm*

453=2<SOH>448=DEUTDEFF500<SOH>447=B<SOH>452=1<SOH>448=GSILGB2XHUL<SOH>447=B<SOH>452=4<SOH>



## Nested Repeating Group Example

- FIX permits nested repeating groups
  - The PartySubGrp is a nested repeating group within the Parties Group
- Example:

NoPartyIDs(tag 453)=2

PartyID(tag 448)=DEU

PartyIDSource(tag 447)=B *note: Bank Identifier Code (BIC) ISO 9362*

PartyRole(tag 452)=1 *note: Executing Firm*

NoPartySubIDs(tag 802)=1

PartySubID(tag 523)=A1

PartySubIDType(tag 803)=10 *note: Securities account number*

PartyID(tag 448)=GSI

PartyIDSource(tag 447)=B *note: Bank Identifier Code (BIC) ISO 9362*

PartyRole(tag 452)=4 *note: Clearing Firm*

NoPartySubIDs(tag 802)=1

PartySubID(tag 523)=C3

PartySubIDType(tag 803)=15632 *note: Securities account number*

```
453=2^448=DEU^447=B^452=1^802=1^523=A1^803=10^448=GSI^447=B^452=4^802=1^523=C3^803=15632^
```



## FIX Checksum

- Sum every byte in the FIX message up to but not including the checksum field
- Take the modulo 256 of the Sum
- Store the value into the CheckSum field (10=nnn)
- Don't forget the trailing `<SOH>` character



# FIX Message Structure

## Standard Header

Identifies message type, message length, sender/destination, sequence number, sending time, etc

```
8=FIXT.1.1<SOH>9=70<SOH>35=A<SOH>49=JPMC1<SOH>56=LSE  
<SOH>34=1<SOH>52=20090310-10:23:01.093<SOH>
```

## Message Body

Contains specific session or application message content

```
98=0<SOH>108=30<SOH>1137=9
```

## Standard Trailer

Contains optional digital signature and the required checksum value

```
<SOH> 10=247 <SOH>
```



## FIX New Order Single Message Example

```
8=FIX.4.2<SOH>9=0235<SOH>35=D<SOH>34=10<SOH>43=N<SOH>49=VENDOR<SOH>50=CUSTOMER<SOH>56=BROKER<SOH>52=19980930-09:25:58GMT<SOH>1=XQCCFUND<SOH>11=10<SOH>21=1<SOH>55=EK<SOH>48=277461109<SOH>22=1<SOH>54=1<SOH>38=10000<SOH>40=2<SOH>44=76.750000<SOH>59=0<SOH>10=165
```

### Header

8=FIX.4.2	Begin String
9=235	Body Length
35=D	MsgType
34=10	MsgSeqNum
43=N	PossDupFlag
49=VENDOR	SenderCompID
115=CUSTOMER	OnBehalfOfCompID
56=BROKER	TargetCompID
52=19980930-09:25:58 GMT	Sending Time

### Body

1=XQCCFUND	Account (optional)
11=10	ClOrdID
21=1	HandInst
55=EK	Symbol
48=277461109	SecurityID (optional)
22=1	IDSource (optional)
54=1	Side
38=10000	OrderQty
40=2	OrdType
44=76.750000	Price (optional)
59=0	TimeInForce (optional)

### Trailer

10=165	Checksum
--------	----------



## Datatypes

- Strings

- Field lengths are not specified within the FIX Specification
- MultipleCharValue
  - string field containing one or more space delimited single character value
  - Example:

ExecInst(Tag 18) Execution Instructions

```
<SOH>18=2 A F<SOH>
```

- MultipleStringValue

- string field containing one or more space delimited multiple character values
- Example:

TradeCondition(Tag 277) Trade Conditions

```
<SOH>277=AV AN A<SOH>
```

- Numbers

- All numeric values represented as ASCII strings
- Integer
- Float

- Dates – variants on YYYYMMDD-HH:MM:SS.sss

- During a leap second insertion, a UTCTimestamp field may read "19981231-23:59:59", "19981231-23:59:60", "19990101-00:00:00"



## Components

- Groups fields that are used together
- Examples
  - Instrument
  - Parties
- Introduced in FIX.4.3
- Does not impact wire format



# Components

359	EncodedHeadline		N	Encoded (non-ASCII characters) representation of the Headline field in the encoded format specified via the MessageEncoding field.
215	NoRoutingIDs		N	Required if any RoutingType and RoutingIDs are specified. Indicates the number within repeating group.
→	<b>216</b>	<b>RoutingType</b>	N	Indicates type of RoutingID. Required if NoRoutingIDs is > 0.
→	<b>217</b>	<b>RoutingID</b>	N	Identifies routing destination. Required if NoRoutingIDs is > 0.
146	NoRelatedSym		N	Specifies the number of repeating symbols (instruments) specified
→	<b>component</b> <Instrument>	<b>block</b>	N	Insert here the set of "Instrument" (symbology) fields defined in "COMMON COMPONENTS OF APPLICATION MESSAGES"
555	NoLegs		N	Number of legs Identifies a Multi-leg Execution if present and non-zero.
→	<b>component</b> <InstrumentLeg>	<b>block</b>	N	Must be provided if Number of legs > 0
711	NoUnderlyings		N	Number of underlyings
→	<b>component</b> <UnderlyingInstrument>	<b>block</b>	N	Must be provided if Number of underlyings > 0
33	LinesOfText		Y	Specifies the number of repeating lines of text specified
→	<b>58</b>	<b>Text</b>	Y	Repeating field, number of instances defined in LinesOfText



## Components

<i>Tag</i>	<i>FieldName</i>	<i>Req'd</i>	<i>Comments</i>
12	Commission	N	
13	CommType	N	
479	CommCurrency	N	For CIV - Optional
497	FundRenewWaiv	N	For CIV - Optional

<i>Tag</i>	<i>FieldName</i>	<i>Req'd</i>	<i>Comments</i>
453	NoPartyIDs	N	Repeating group below should contain unique combinations of PartyID, PartyIDSource, and PartyRole
→	448 PartyID	N	Used to identify source of PartyID. Required if PartyIDSource is specified. Required if NoPartyIDs > 0.
→	447 PartyIDSource	N	Used to identify class source of PartyID value (e.g. BIC). Required if PartyID is specified. Required if NoPartyIDs > 0.
→	452 PartyRole	N	Identifies the type of PartyID (e.g. Executing Broker). Required if NoPartyIDs > 0.
→	<i>Start of Component block, expanded in line &lt; PtysSubGrp &gt;</i>		
→	802 NoPartySubIDs	N	
→	→ 523 PartySubID	N	
→	→ 803 PartySubID Type	N	
→	<i>End of Component block, expanded in line &lt; PtysSubGrp &gt;</i>		



## FIXML Schema

- Schema published for FIX.4.4, FIX.5.0, FIX.5.0SP1, FIX.5.0SP2
- *FIX Messages* and *FIX Components* represented as XML Elements  
`<TrdCaptRpt />`
- *FIX Fields* represented as XML Attributes  
TradeReportID(Tag 571) becomes @RptID
- Uses *FIX Field Abbreviated Names*

35=D<SOH>1=A<SOH>

9 bytes for FIX

`<Order Acc="A" />`

16 bytes for FIX



## FIXML Design Rules

- Abbreviations to reduce message size
  - Price = Px, Currency = Ccy, etc.
- *FIX Messages* are implemented as XML Elements
- Non- repeating *FIX Fields* are implemented as XML Attributes to the *FIX Message XML Element*
- *FIX Components* are implemented as XML Elements
  - *FIX Fields* within *FIX Components* are implemented as XML Attributes
- *FIX Repeating Groups* must be *FIX Components*
- *FIX Datatypes* are mapped to the closest XML Schema Datatype
- *FIX Fields* are identified by their *FIX Field Abbreviated Name*



## FIXML Field Contextual Abbreviations

- *FIX Fields* can have a *FIX Field Contextual Abbreviated Name* in certain cases
  - *FIX Fields* often have prefixes that associate the field to one or more messages within a *FIX Message Category*
  - Within that *FIX Message Category* the *FIX Field* can be assigned a *FIX Field Contextual Abbreviated Name*
  - Examples:
    - **ListID(Tag 66)** Standard Abbreviation: **ListID** Contextual Abbreviation: **ID** in **Allocation** Category
    - **AllocID(tag 70)** Standard Abbreviation: **AllocID** Contextual Abbreviation: **ID** in **Allocation** Category
- Multiple *FIX Fields* can share the same *FIX Field Abbreviated Name*
  - Examples:
    - **ID** for PartyID(tag 448), NestedPartyID(tag 524), Nested2PartyID(tag 757), ...
    - **R** for PartyRole(tag 452), NestedPartyRole(tag 538) Nested2PartyRole(tag 759)...
    - **Src** for PartyIDSource(tag 447), NestedPartyIDSource(tag 525) ,. ...



## Tag = Value compared to FIXML

- Tag = Value

```
8=FIX.4.1<SOH>9=119<SOH>35=6<SOH>49=BRKR<SOH>56=INVMGR<SOH>34=236<SOH>52=19980604-
07:58:48<SOH>23=115685<SOH>28=N<SOH>55=SPMI.MI<SOH>54=2<SOH>27=200000<SOH>44=10100.000000
<SOH>25=H<SOH>10=159<SOH>
```

Length 136 bytes

- FIXML

```
<FIXML>
  <IOI IOIID="115685" TransTyp="N" Side="2" Qty="200000"
Px="10100.000000" QtyInd="H">
  <Hdr SeqNum="236" SndgTm="1998-06-04T07:58:48">
    <Snd ID="BRKR" />
    <Tgt ID="INVMGR" />
  </Hdr>
  <Instrmt Sym="SPMI.MI" />
</IOI>
</FIXML>
```

Length 248 bytes

# FIX Session Layer

**FIX**PROTOCOL  
INDUSTRY-DRIVEN MESSAGING STANDARD™



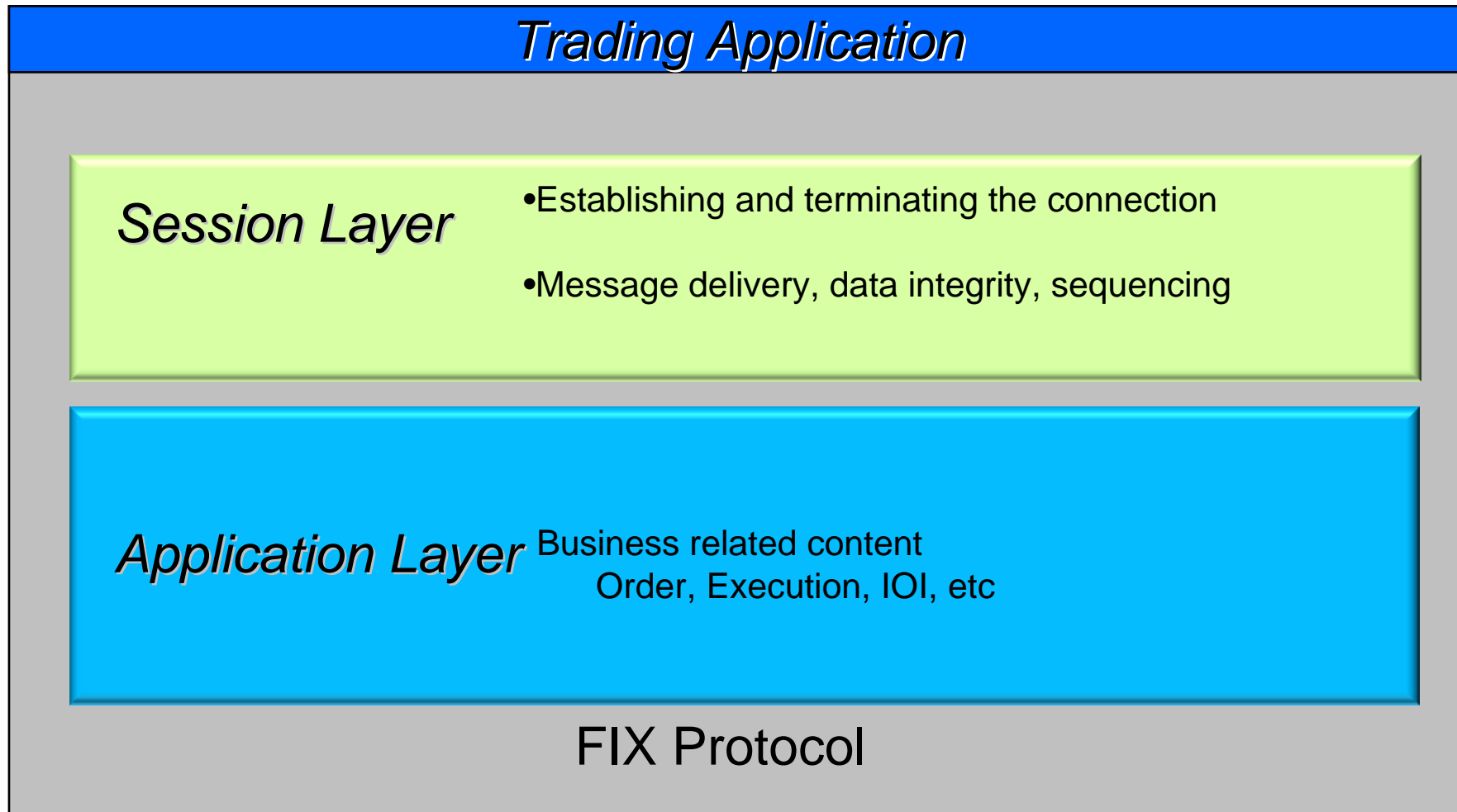


## Introduction to the FIX Session Layer - Agenda

- FIX Session Layer (FIX 4.0 - 4.4 and FIXT.1.1)
  - Overview
  - FIX Engine Anatomy and Functions
  - Session Level Messages and Usage
- FIXT.1.1
  - Extension Packs and Service Packs
  - Transport Independence
  - Application Version Independence

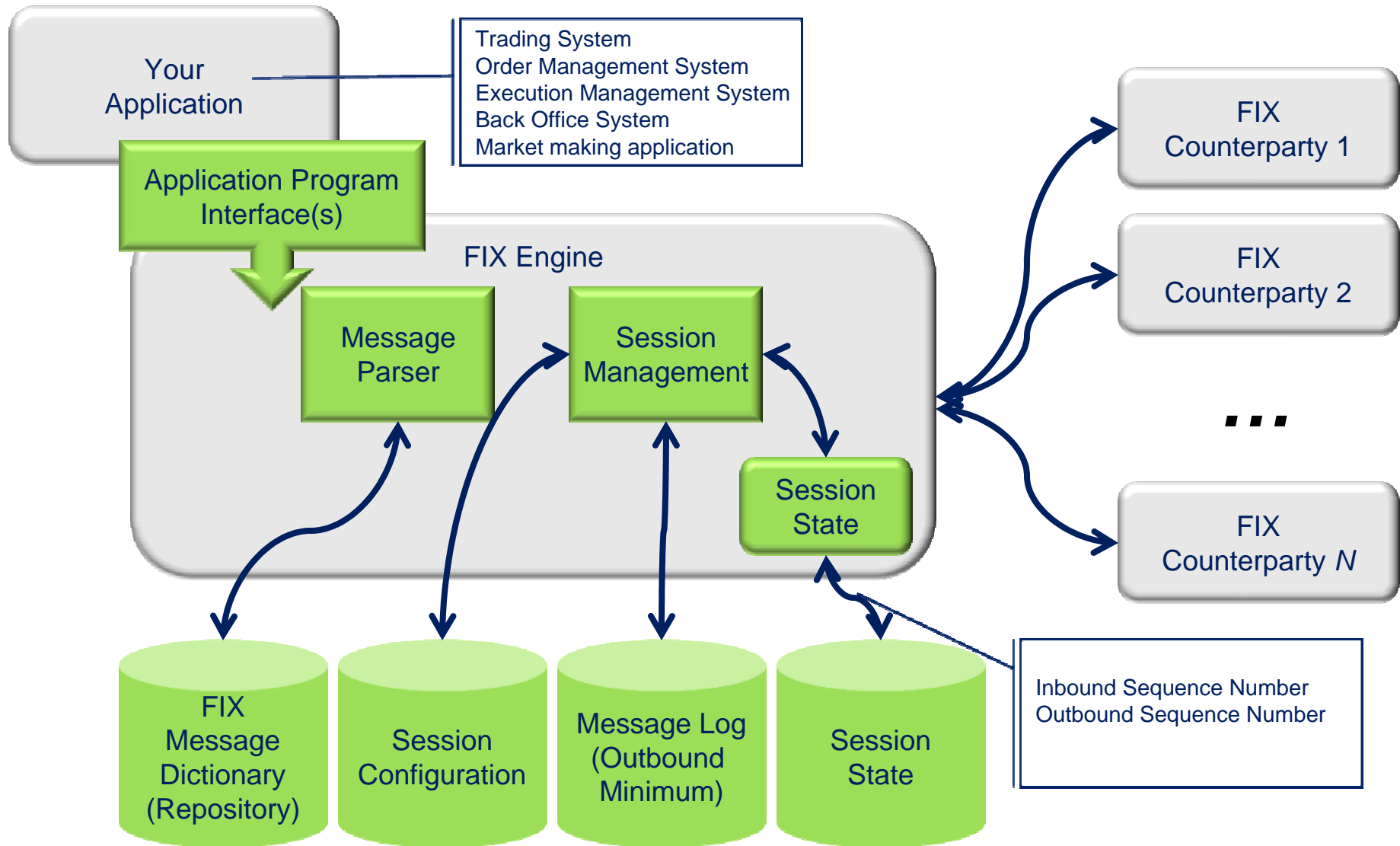


## FIX Session and Application Layers





# Anatomy of a FIX Engine





## FIX Engine - Key Functions

- Session initiation
  - Get configuration details from Session Configuration (e.g. IP address, TCP Port, CompIDs, etc)
  - Determine last inbound/outbound sequence numbers or set to 1 if first session of the day
  - Connect to internal business message “handlers” via an API
  - Connect to FIX session counterparty
  - Send Logon and perform Logon handshake
- Continuous functions
  - Service inbound FIX messages
    - Decrypt (optional), parse, and safe-store all messages
    - Respond to session-level messages
    - Convert and forward business messages to “handler” via an API
    - Validate seq num, send Resend Request if gap detected
  - Service outbound requests from internal “handlers”
    - Construct as FIX message, encrypt (optional), safe-store, and send over FIX session to counterparty
  - Administrative functions
    - Send Heartbeats, Test Requests, system status
    - Logout at session “end” time



## Session Level Functions

- Initiating a session (Logon)
- Message Validation
- Message Sequencing
- Heartbeat
- Ending a session (Logout)



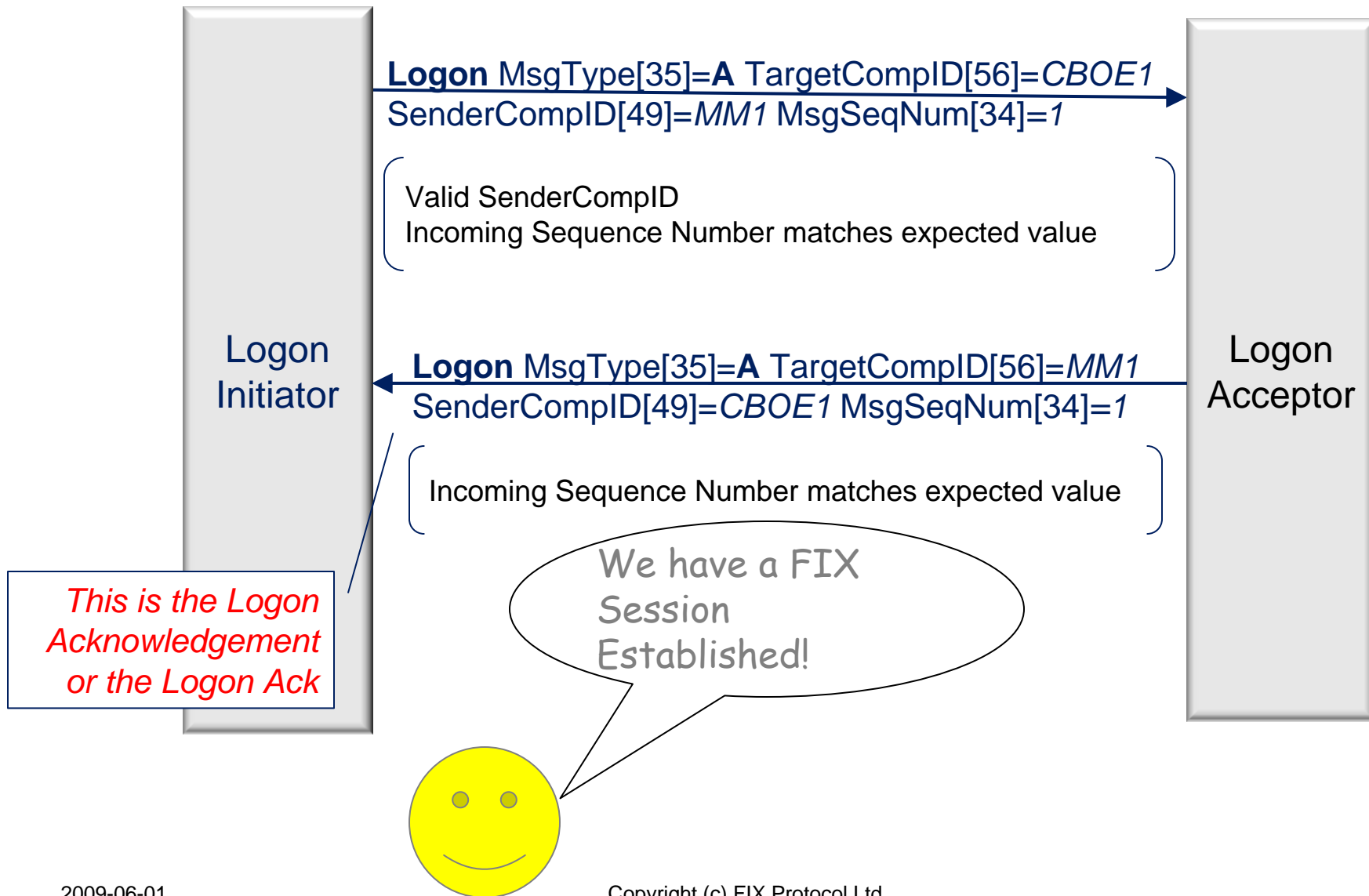
## Initiating a Session (Logon MsgType[35]=A)

- Buy-side firm initiates (the *Initiator*) a connection to a Sell side (the *Acceptor*) firm using the Logon Session Level Message

```
BeginString[8]=FIX.4.2<SOH>  
BodyLength[9]=92<SOH>  
MsgType[35]=A<SOH>  
SenderCompID[49]=BUYSIDE1<SOH>  
TargetCompID[56]=SELLSIDEA<SOH>  
SenderSubID[50]=MSO:MSO<SOH> [Optional]  
TargetSubID[57]=TEST<SOH> [Optional]  
MsgSeqNum[34]=106<SOH>  
PossDupFlag[43]=N<SOH> [Optional]  
SendingTime[52]=20010822-13:06:42<SOH>  
EncryptMethod[98]=0<SOH>  
HeartBtInt[108]=30<SOH>  
Checksum[10]=021<SOH>
```



## Nominal Login Scenario Start of Day





## Message Validation

- FIX Engines validate that messages are properly formed and will reject the message using a Session Level Reject message if the message is invalid
- Validations can include but are not limited to:
  - Message structure (Tag=Value[SOH]) and CheckSum[10]
  - Field order (e.g. BeginString[8] is first, Header / Body / Trailer order, repeating groups)
  - MsgType[35] is valid and supported
  - All required fields are specified
  - Field format matches specified data type
  - Tags specified without values (e.g. Tag=[SOH] is not allowed)
  - Proper SenderCompID[49] and TargetCompID[56]



## Message Sequencing

- The FIX Session guarantees that messages are delivered in the order in which they are sent
- A FIX Session can be defined as
  - *“a bi-directional stream of ordered messages between two parties within a continuous sequence number series”*
- Each FIX Engine is expected to keep track of two sequence numbers
  - The **Incoming Sequence Number** expected on inbound messages received from the counterparty
  - The **Outgoing Sequence Number** to be sent to the counterparty
- The FIX session layer provides a method for recovery of lost messages, or messages received out of order
  - FIX applications can resynchronise sequence numbers between engines using the FIX Session Layer

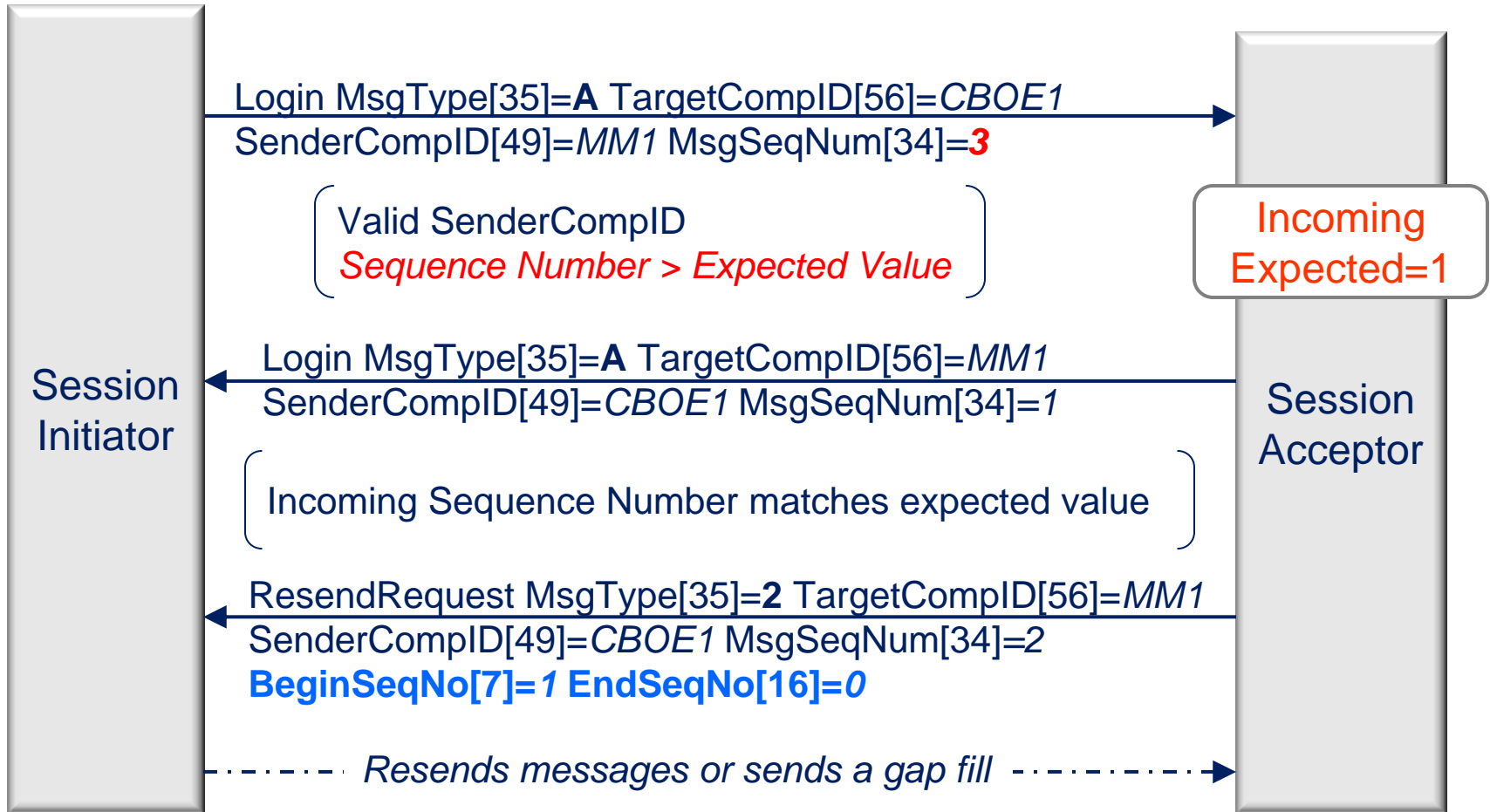


## Resend Request (MsgType[35]=2)

- Used to request the counterparty to resend a set of FIX messages
- Tell the counterparty the range of messages that you did not receive
  - BeginSeqNo[7]
  - EndSeqNo[16]
- Can ask for a specific range, or can request all messages from BeginSeqNo[7] to infinity (recommended)
  - For FIX 4.2 and higher, infinity is EndSeqNo[16]=0
  - For FIX 4.0 and 4.1, infinity is EndSeqNo[16]=999999
- Acceptable responses:
  - Resend the messages in question with PossDupFlag[43]=Y, or
  - Send Sequence Reset – Gap Fill (MsgType[35]=4 with GapFillFlag[123]=Y)

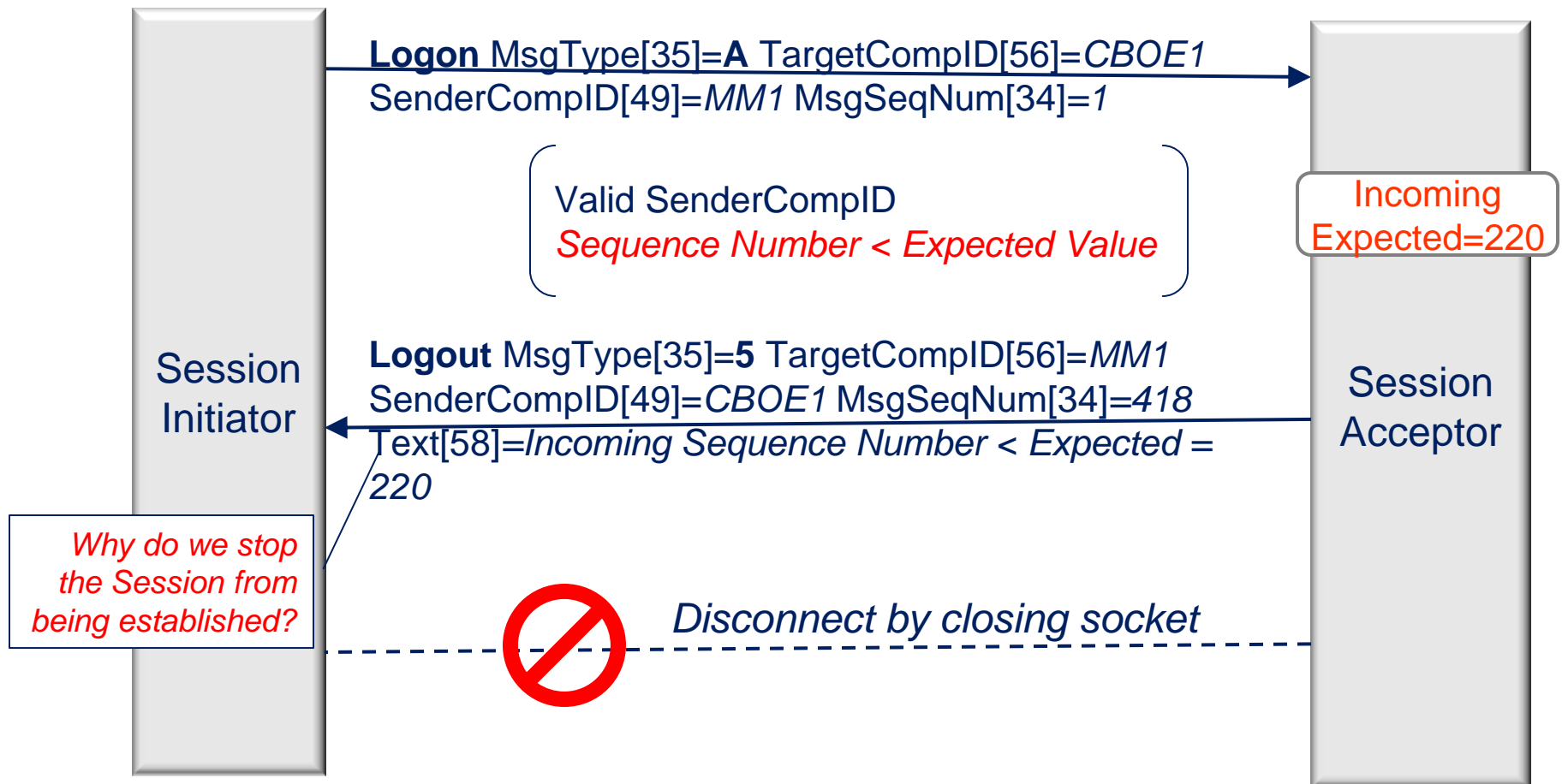


# Exceptional Login Scenario Sequence Number too High





# Exceptional Login Scenario Sequence Number to Low



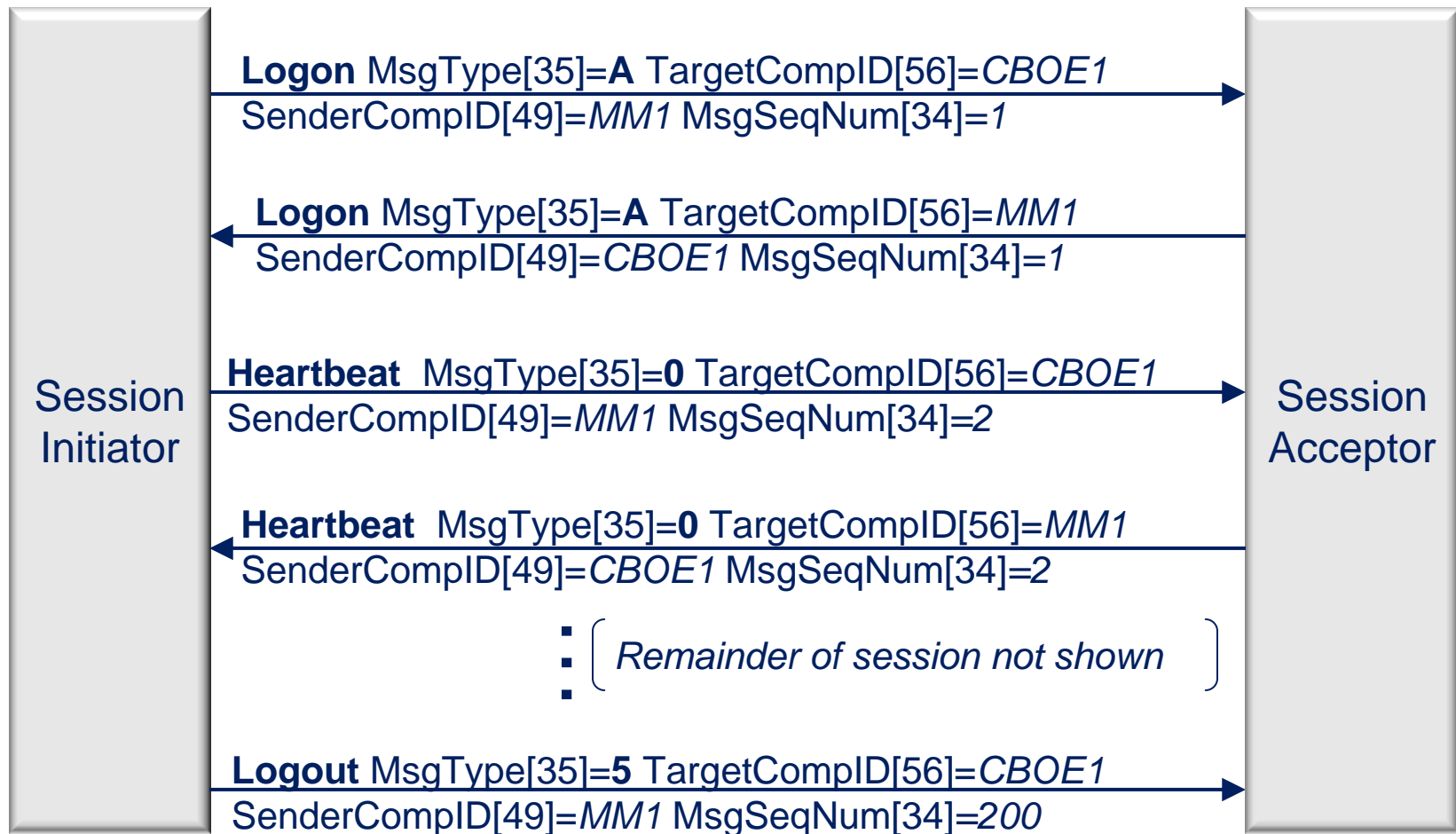


## Sequence Reset (MsgType[35]=4)

- In order to reduce unnecessary communication, FIX permits firms to skip or gap fill over administrative messages (such as heartbeats and test requests)
- This can also be used to skip over stale messages such as orders
- This is done using the **SequenceReset** Message
  - GapFillFlag[123]
    - GapFillFlag[123]=Y must be processed in sequential order, and is recommended for normal message recovery
    - GapFillFlag[123]=N forces a change in sequence number and is intended for emergencies or manual intervention
  - NewSeqNo[36]
    - This is the next message the recipient should expect



# Heartbeat (MsgType[35]=0) lets each side know the connection is active





## How often should we send Heartbeat messages?

- Configurable
  - Originally many sites would use 30 seconds
  - Many firms are using 15 seconds
- “Negotiated” at Logon time
  - HeartBtInt [108] – integer value in seconds
  - Both sides should agree upon Heartbeat Interval
- Optimisation
  - Only send Heartbeats when the session is inactive

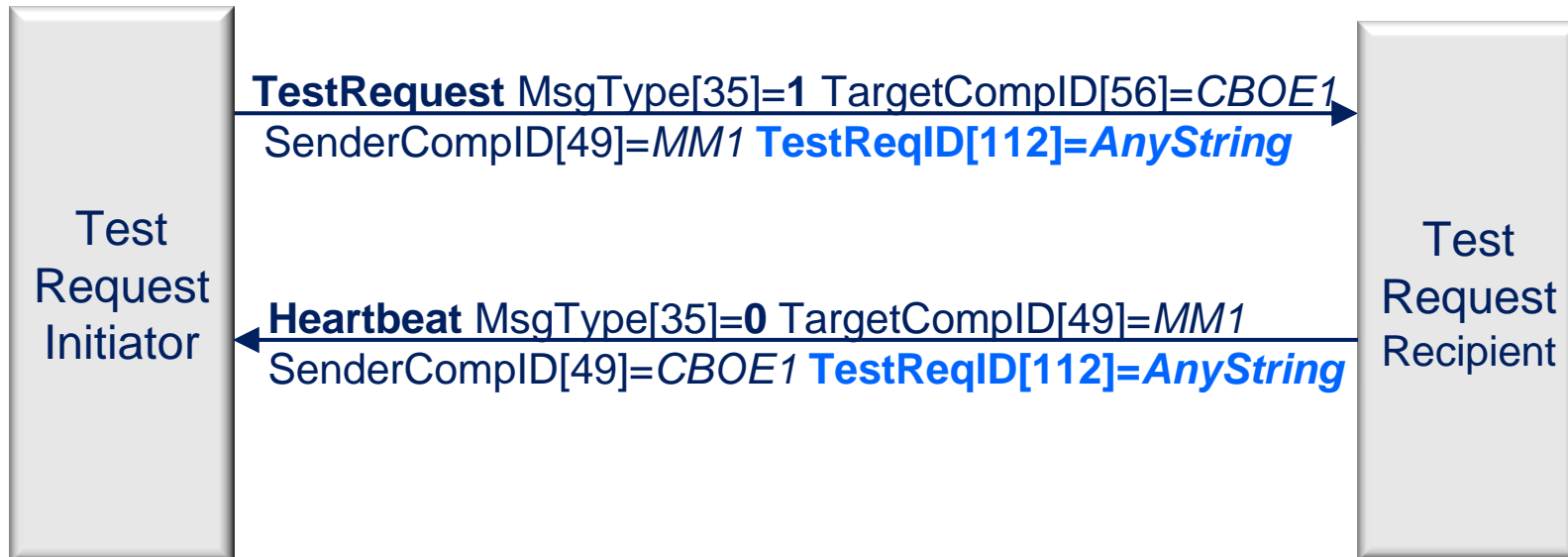


## Test Request (MsgType=1)

- Either party on the FIX connection can send a Test Request message at any time during the FIX Session
- 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[112] of the Test Request
- A Test Request is sent if no messages are received in more than HeartBtInt[108] seconds
- Failure to respond to the Test Request causes a Logout and disconnection



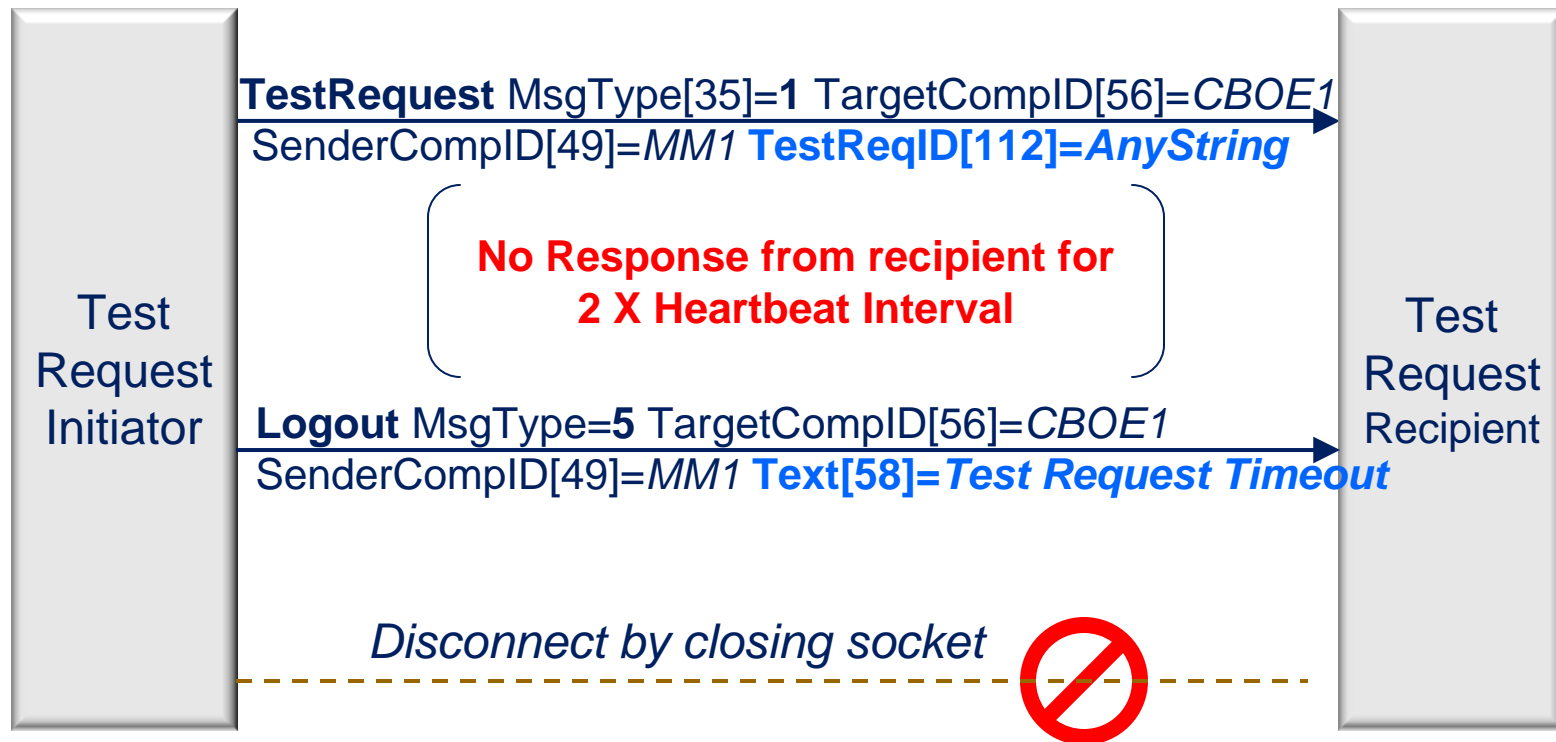
## Normal Test Request Scenario





# Exception Test Request Scenario

## No response from counterparty





## Ending a Session(Logout MsgType=5)

- Each side sends a logout message

8=FIX.4.2<SOH>

9=82<SOH>

35=5<SOH>

49=CBOE1<SOH>

56=MM1<SOH>

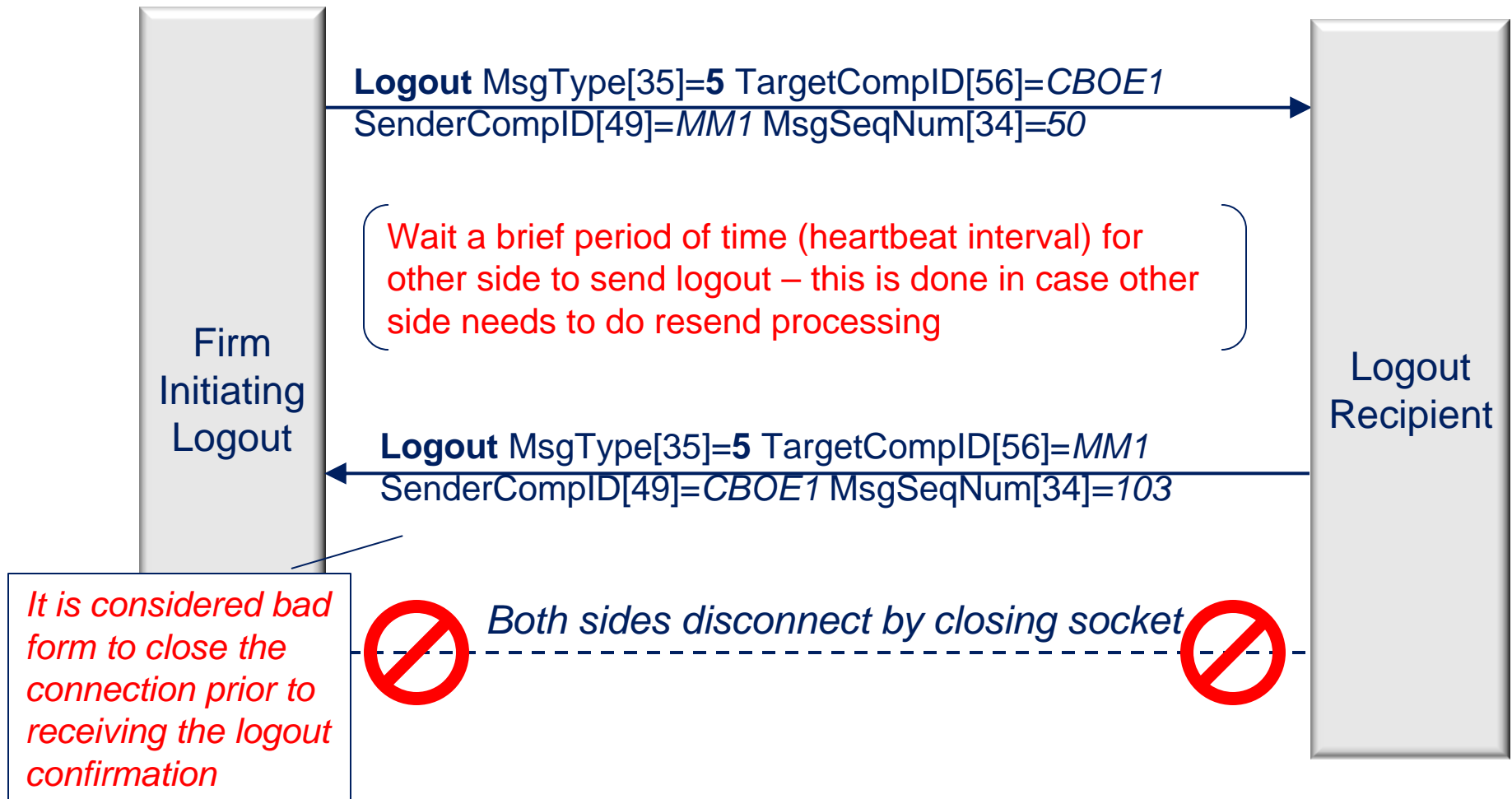
34=483<SOH>

52=20010822-14:05:34<SOH>

10=176<SOH>



# Normal Logout Processing





## FIX 5.0 and FIXT.1.1 - Key Changes

- Actually 2 Protocols:
  - FIX 5.0 – Application level protocol, including support for new business functionality
  - FIXT.1.1 – Session level protocol
- Transport Independence
- Application Version Independence
- First release driven fully from the Repository
  - XML representation of the FIX protocol
  - Machine-readable, reducing cost of building validation rules into FIX engines
  - The Repository generates the Word document specification, FIXimate, and the FIXML Schema.
- First release where all documentation was published at the same time
- First release using the new ‘Service Pack’ architecture
  - Users prepare Extension Packs and GTC approves them
  - GTC packages Extension Packs together into Service Packs
  - GTC can release Extension Packs independent of Service Pack releases, allowing rapid adoption of proposed changes



## Transport Independence

- FIX 5.0 is the first version to de-couple FIX application messages from the FIX session
- FIXT.1.1 is a distinct specification that does not change with FIX version
- FIX 5.0 can be used over other transports:
  - MQ Series
  - Web Services
  - HTTP
  - Message Bus
  - FIX Session Layer (FIXT.1.1)
- Allows greater return on investment on existing FIX investments
- Reduces spend to add new FIX functionality
- Offers opportunity to choose appropriate transport for the circumstances



## Application Version Independence

- Application level messages from any FIX version can be sent over FIXT.1.1
- Addresses concerns about the number of FIX versions by allowing firms to mix and match, all on the same session
- Ability to add incremental functionality with incremental development, e.g. you can now add FIX 4.4 Allocations to an existing FIX 4.2 Orders and Executions system as follows:
  - Replace FIX 4.2 session with a FIXT.1.1 session defaulted to FIX 4.2
  - Write code to handle FIX 4.4 Allocations
  - FIX 4.2 Order and Execution system does not change.
- Without FIXT.1.1:
  - Additional cost, time, and testing effort to upgrade Orders and Executions to FIX 4.4
  - Additional development and certification for all counterparties to upgrade their Orders and Executions systems to FIX 4.4



## FIXT.1.1 and Application Version Independence

- Application Version controlled by new fields:
  - DefaultAppVerID[1137] – Required at Logon
  - AppVerID[1128] – Appears in Header, overrides DefaultAppVerID
  - Additional fields added to support Extension Packs and custom application versions
- Full support for multiple application versions requires significant rework of a FIX engine
  - See FIXT.1.1 Specification
- Many users only require support of FIX 5.0 or a FIX 5.0 Service Pack
- Approach:
  - Use FIXT.1.1 to support only one version
  - Specify the version using DefaultAppVerID[1137]
  - The Engine and application only need to be aware of one FIX version and data dictionary.

# FIX Semantics

**FIX**PROTOCOL  
INDUSTRY-DRIVEN MESSAGING STANDARD™





## Why talk about FIX Semantics?

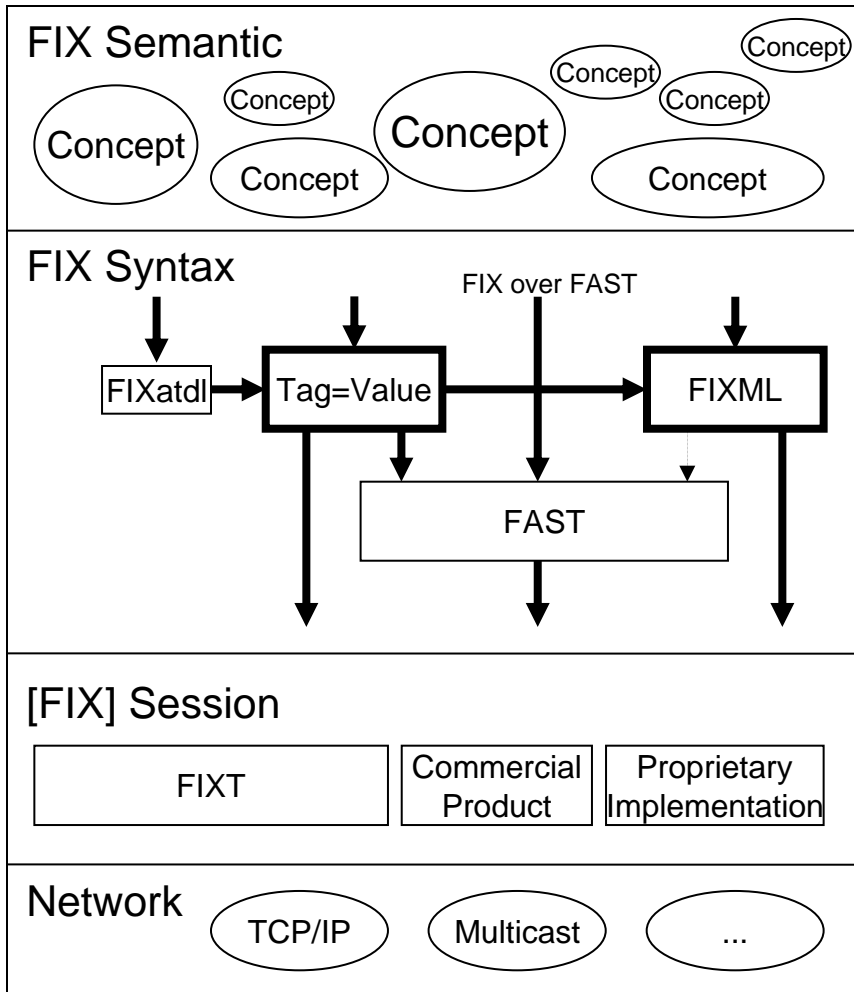
Semantics are the main cost driver for the development of applications that process interface messages.

- Differences in semantics can require complex translations between different representations (not the syntax level)  
→ Complex translations add effort and latency
- Differences in semantics can require keeping state at a gateway level by means of a database  
→ Keeping state adds effort and latency

**→ FIX defines standard semantics for the industry to reduce development cost and latency**



# FIX Semantic, Syntax, Session



## PRINCIPLES

- FIX semantics layer (application) consists of one or more concepts for each business functionality
- FIX syntax layer represents the encoding of the FIX semantics
- FIX syntax can start with either FIXatdl, Tag=Value or FIXML
- Tag=Value can optionally be converted to FAST
- Tag=Value can be converted to FIXML
- One of Tag=Value, FIXML or FAST is provided to the session layer
- Session layer (transport) includes non-FIX protocols
- FIX versions prior to FIX 5.0 use the standard FIX Transport Protocol
- TCP/IP is typically used for transactions
- Multicast is well suited for data distribution

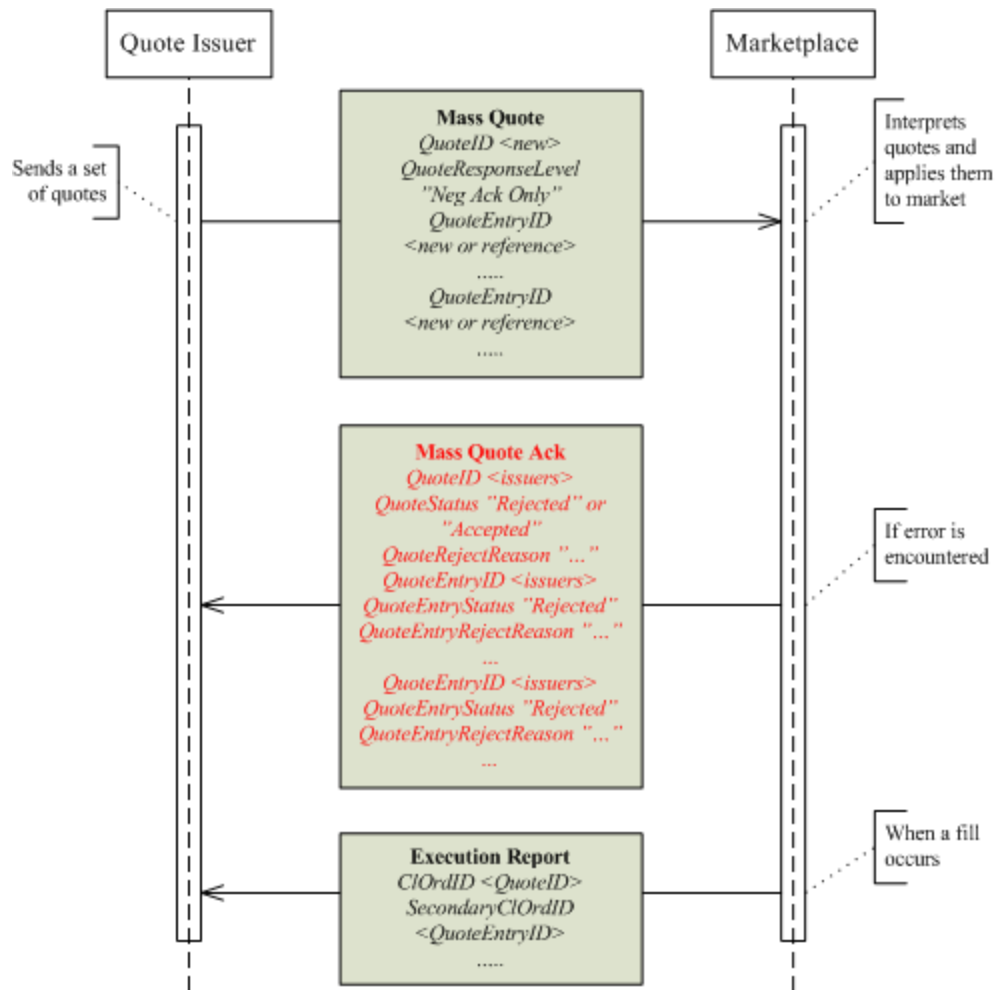


## FIX Semantics – Key Elements

- **Transaction Model**
  - What is the purpose of a given message?
  - What is the message flow for a given business transaction?
  - How many messages are exchanged and in which sequence?
- **State Model**
  - Which states can entities such as orders and quotes have?
  - Which state transitions are possible and how are they triggered?
- **Identification Model**
  - How are orders, quotes and trades identified and by whom?
  - How is the ISO identification standard integrated?
- **Key Fields**
  - Which fields allow to tie requests and responses together?
  - Which fields drive the behavior of a message?
  - How are application level errors reported back?



# FIX Semantics – Transaction Model: Mass Quote





# FIX Semantics – Transaction Model: Order Entry

## *1.1.b – Immediate or Cancel order that cannot be immediately hit completely*

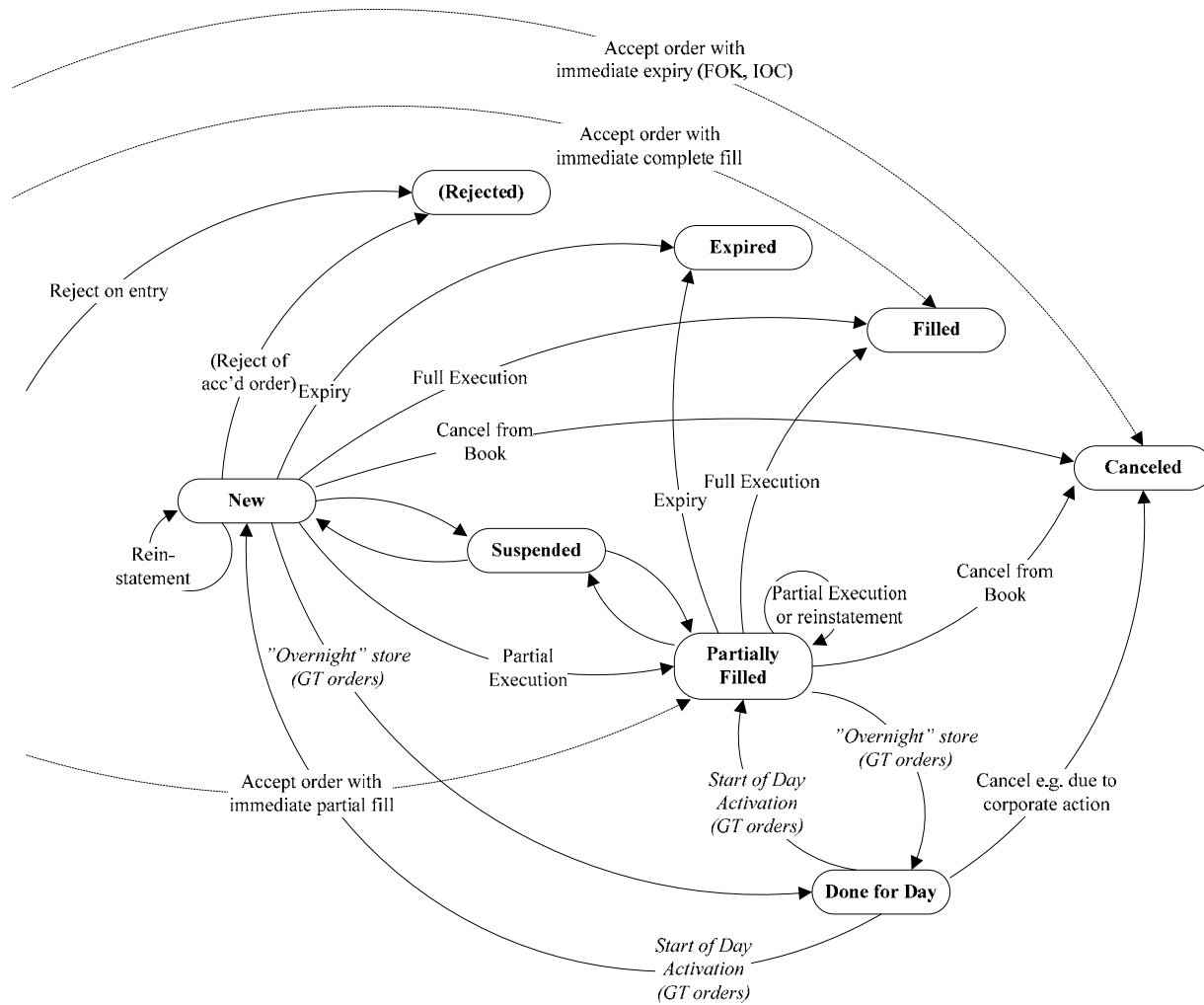
<u>Time</u>	<u>Message Received</u> (ClOrdID, OrigClOrdID)	<u>Message Sent</u> (ClOrdID, OrigClOrdID)	<u>Exec Type</u>	<u>OrdStatus</u>	<u>Order Qty</u>	<u>Cum Qty</u>	<u>Leaves Qty</u>	<u>Last Qty</u>	<u>Comment</u>
1	New Order(X)				10000				Order is IOC
2		<i>Execution(X)</i>	<i>Rejected</i>	<i>Rejected</i>	<i>10000</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>If order is rejected by sell-side (broker, the exchange, ECN)</i>
2		Execution(X)	New	New	10000	0	10000	0	If messages are not bundled
3		Execution(X)	Trade	Partially Filled	10000	1000	9000	1000	Execution for 1000
4		Execution(X)	Canceled	Canceled	10000	1000	0	0	If order cannot be immediately hit completely
5	New Order(Y)				10000				Order is IOC
6		<i>Execution(Y)</i>	<i>Rejected</i>	<i>Rejected</i>	<i>10000</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>If order is rejected by sell-side (broker, the exchange, ECN)</i>
6		Execution(Y)	Trade	Canceled	10000	1000	9000	1000	If message bundling is being used and execution of 1000 occurs immediately upon hitting the book

➔ **FIX supports more than a single concept wherever this is warranted by the user group (buy-side/sell-side vs. exchanges) or asset class.**



# FIX Semantics – State Model for Exchanges

**FIX Field**  
**OrdStatus**  
**(39)**





## FIX Semantics – Identification Model (1)

- Orders (single leg, multi-leg)
  - ClOrdID as entity and message identifier of order issuing side
  - Concept of ClOrdID as message identifier requires a new identifier for transactions on existing orders and a reference (OrigClOrdID)
  - OrderID as entity identifier of order receiving side
- Quotes (single quote, mass quote)
  - Entities typically identified implicitly by instrument (multiple fields)
  - Explicit quote identifier (QuoteID) only needed if quote issuer is allowed to have multiple quotes per instrument (derivatives: per series)
  - Mass Quote identifier (QuoteID, QuoteSetID, QuoteEntryID)
- Fills/Trades
  - Execution identifier (ExecID, FillExecID) for fills on orders and quotes
  - Trade identifier (FirmTradeID, TradeID, SideTradeID) for entity level
  - Trade Report identifier (TradeReportID, SideTradeReportID) for message level



## FIX Semantics – Identification Model (2)

### ● Instruments

- Equities: Symbol or SecurityID + SecurityIDSource
- Futures: add CFICode and MaturityMonthYear
- Options: add StrikePrice, optionally also PutOrCall
- SecurityIDSource=M (marketplace-assigned) for synthetic identifiers
- Lists: SecurityListID, e.g. for industry classifications
- Groups: SecurityGroup, e.g. for exchange specific partitions

### ● Markets and Market Segments

- MarketID defined as MIC (ISO 10383)
- MarketSegmentID defined as arbitrary string

### ● Trading Sessions

- TradingSessionID: Day (1), Half-Day (2), Morning (3), Afternoon (4), Evening (5), After-hours (6)
- TradingSessionSubID: Pre-Trading (1), Opening (2), Trading (3), Closing (4), Post-Trading (5), Intra-day Auction (6), Quiescent (7)



## FIX Semantics – Identification Model (3)

- ISO Identification in FIX
  - Instruments: ISIN (ISO 6166), CFI Code (ISO 10962)
  - Parties: BIC (ISO 9362)
  - Marketplaces: MIC (ISO 10383)
  - Currencies: 3 characters (ISO 4217)
  - Countries: 2 characters (ISO 3166-1)
  - Languages: 2 characters (ISO 639-1)
  - Timestamps: UTC (ISO 8601)



## FIX Semantics – Key Fields

- Which fields tie requests and responses together?
  - FIX offers explicit, distinct message identifiers in the body of the message
  - xxxReq[uest]ID identifies a request message and is returned in the response
  - xxxRptID or xxxReportID identify (unsolicited) reports
  - Exception: order handling (request: ClOrdID, report: ExecID)
  - Application Sequencing offers generic field ApplSeqNum for reports
  - xxxAck messages echo input fields to allow drop-copies to other sessions
- Which fields drive the behavior of a message?
  - Fields like ExecInst, TradeReportTransType, MDUpdateAction tell the recipient what to do with the entity described in the message
  - Fields like ExecType, OrdStatus, TrdRptStatus tell the sender what the recipient has done with the entity described in the message
- How are application level errors reported back?
  - xxxStatus (with exceptions), xxxResult, xxxRej[ect]Reason confirm success or failure and explain why an application request was not accepted
  - Text field is a catch-all for free text information that can also be used to convey errors



## FIX Semantics – Example: Order Quantity (1)

How do I convey quantity information when I want to modify an existing order?

- FIX field: OrderQty, data type int
- Option 1: Delta quantity **Non-FIX**
  - Add given amount to current order quantity at the receiver
- Option 2: Remaining quantity **Non-FIX**
  - Ensure order has given amount after modification by the receiver
- Option 3: Total quantity **FIX**
  - Calculate offset to previous total quantity and apply delta

➔ Only option 3 is compliant with FIX semantics



## FIX Semantics – Example: Order Quantity (2)

Which message(s) do I use when I want to increase or decrease the quantity of an existing order?

- Option 1: Two different messages **Non-FIX**
  - Increase quantity: use Order Cancel/Replace
  - Decrease quantity: use Order Cancel with a user defined quantity field
- Option 2: Single message **FIX**
  - Use Order Cancel/Replace to increase or decrease the quantity
  - Use Order Cancel only if you decrease the quantity to zero (delete order)

➔ Option 2 is compliant with FIX semantics\*

\*FIX 4.0 supported option 1 but the semantic was changed with the introduction of FIX 4.1 in April 1998



## FIX Semantics – Example: Market Data

How do I convey order book depth information which is aggregated per price level?

- Option 1: Two different messages **FIX**
  - Use MarketDataSnapshotFullRefresh to provide price and quantity for all available price levels (recipient can simply replace his own copy)
  - Use MarketDataIncrementalRefresh to instruct the recipient to make specific changes to his copy (add, change, delete price level bid or ask)
- Option 2: Single message **Non-FIX**
  - Use MarketDataSnapshotFullRefresh to provide price and quantity for all price levels where something has changed (recipient to modify information only for price levels included in the snapshot)

➔ Option 1 is compliant with FIX semantics



## FIX Semantics – Reference Data with FIX 5.0 SP1

- FIX 5.0 SP1 significantly enhanced the area of Reference Data
  - Market (segment) level messages (MarketDefinition[Request][UpdateReport])
  - Possibility to define trading rules on the level of a market (segment), trading session or individual instrument
    - Rules defining available order types, execution instructions, time in force values, matching algorithms, market data feeds
    - Rules defining ticks, lot types, price types, price limits, minimum/maximum order quantities
    - Rules defining increments and exercise styles per strike range as well as maturity rules
- FIX 5.0 SP1 enabled modeling of Market Hierarchies
  - A market at the top level is always a marketplace defined by a MIC
  - There can be zero or more market segments within a market representing market specific groupings of instruments which are ordered as a hierarchy (via field ParentMktSegmID)
  - A market segment is a generic term that needs to be defined by the marketplace
- FIX 5.0 SP1 introduced a concept for propagation of Trading Rules
  - A trading rule defined at a higher level applies to all entities at lower levels
  - A trading rule defined at any level overrides a trading rule defined at a higher level
  - Implementation specific rules of engagement need to be well designed and defined

complex



## FIX Semantics – Reference Data with FIX 5.0 SP2

- FIX 5.0 SP2 enabled modeling of Parties and Risk Limits Reference Data
  - Two new messages:
    - PartyDetailsListRequest [type 'CF']
    - PartyDetailsListReport [type 'CG']
  - Current modes of operation:
    - Request / Response
    - Unsolicited reports
  - Can be used to model:
    - A single party or list of parties
    - The relationships between parties, e.g. Clears for, Trades through, Member of
  - Uses include:
    - Entitlements (which Traders can trade which Accounts, clearing relationships....)
    - Highly flexible risk limits, including multiple simultaneous limits
      - Limit types: Gross, Net, Exposure, Long, Short
      - Scope based on Party, Relationship between Parties, or Instrument