FIX Protocol

Agenda - Part 1

- FIX Protocol Introduction
  - Overview
  - Usage / Players
  - Message Types
  - Message Format
  - Communication Model
  - Supported Product Classes
  - Message Categories

Agenda - Part 2

- Trade Messages
  - New Order
  - Execution Report
  - Order Cancel/Replace
- Trade Scenarios

Agenda - Part 3

- Technical Implementation (quickfixj)
  - Application
  - Session
  - Configurations
- Case study
  - Sequence number reset

Part 1

- FIX Protocol Introduction
  - Overview
  - Usage / Players
  - Message Types
  - Message Format
  - Communication Model
  - Supported Product Classes
  - Message Categories

What is FIX?

- FIX -Financial Information Exchange
- FIX Protocol is an industry driven messaging standard for exchange of trading related information between financial institutions.
- FIX Protocol specification provides format for electronic messages and communication model
- It is widely used protocol in the Financial Markets Industry today

Industry Players & Usage

- Exchanges
  - Use FIX to receive trades from their members and send executions back and other trading related messages
- Buys-side firms
  - Use FIX to send and receive pre-trade, trade and post-trade messages to and from Sell-side firms
- Sell-side firms
  - Use FIX to receive and send pre-trade, trading, post-trade messages from and to buy-side firms
  - Use to communicate with Exchanges and other OTC markets

Supported Product Classes

- Equities
- Fixed Income
- Derivatives (Options, Futures etc)
- FX etc.

Message Categories

- Admin Messages
  - Used to maintain the different aspects of FIX session (connection)
- Application Messages
  - Messages used for transmission business messages

Admin Messages

- Logon – Client Authentication Message
- Logout – Normal Termination of Session
- Heartbeat- Used to check communication link between two parties
- Test Request – used to test the health of the communication link
- Resend Request – Request to retransmit the certain application messages
- Reject (Session Level) – session level validation failure (different from application level validation)
  - Example invalid version, msgtype etc
  - RejectReason is populated with error info
- Sequence Reset/Gap Fill
  - In case of communication problems missing messages recovered or sequence is reset to ignore the missing messages

Application Messages

- Pre-trade messages
  - Quotes, News, Email, Market Data, Security Info etc
- Trade Messages
  - Single Orders, Basket/List Orders, Multi-leg orders, Executions, Order Cancel, Cancel/Replace, Status etc
- Post-Trade Messages
  - Allocations, Settlement Instructions, Positions Mgmt etc

FIX Message Format

- FIX is a platform independent protocol
- Message contains 3 parts:
  - Header
  - Body
  - Trailer
- Each field is a tag-value pair
  - <tag>=<Value>
  - Eg: 55=IBM (Symbol=IBM)
- All fields are terminated by delimiter character - “SOH“ (#001) (in print ‘^’ is used)
- 9=0235^35=D^34=10^43=N^

FIX Message Format

- Tag
  - FIX uses predefined Tags
  - Each Tag represent the specific field
  - Each tag is given a predefined number
  - FIX Field dictionary provides the list of Fields and corresponding Tag numbers (Supplied with Spec)
  - User Defined Tags: range of 5000 to 9999
- Value
  - Values represent the value of the Tag assigned to
  - Supported Data Types are: int, float, char, time, date, data, string

FIX Message Format

- All messages start with “8=FIX.x.y”
  - Indicates the FIX version of the message being transmitted
  - Useful to support multiple versions
- All messages terminate with “10=nnn<SOH>“
  - nnn represents the Checksum of the data
  - Checksum is the sum of all the binary values in the message
  - Checksum helps to identify the transmission problems

Sample Message

- New Order Single Message

8=FIX.4.2^9=0235^35=D^34=10^43=N^49=VENDOR^50=CUSTOME^
56=BROKER^52=19980930-09:25:58^
1=U00611^11=10^21=1^55=SPY^48=277461109^22=1^54=1^
38=10000^40=2^44=76.750000^59=0^10=165

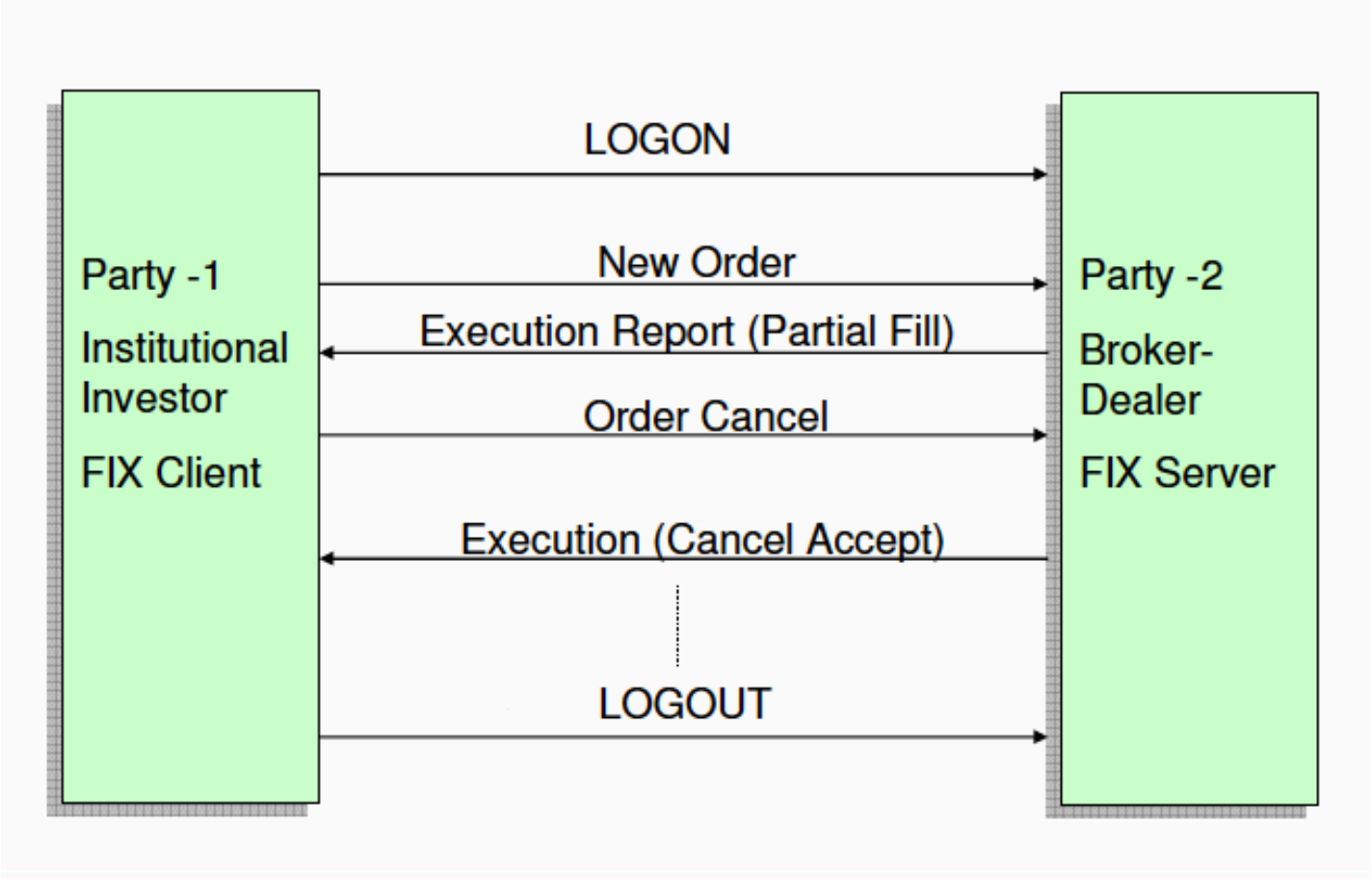
Communication Model

- Session based communication
- Session is communication between two parties
- Initiator / Client
  - party who initiates the communication
- Acceptor /Server
  - party who receives connection request from Initiator
  - Server validates client request using login message

FIX Session

- FIX is a session protocol
  - Each session maintains the bi-directional messages between two parties
  - Session can spread across multiple physical connections
  - Session is maintained using sequence number
  - Both parties rely on sequence numbers to maintain the orderly communication
  - Every new session starts with sequence number 1
  - Missing messages are re-transmitted with bi-lateral agreement between both parties

Sample Flow


Party -1
Institutional Investor
FIX Client
LOGON
New Order
Execution Report (Partial Fill)
Order Cancel
Execution (Cancel Accept)
LOGOUT
Party -2
Broker-Dealer
FIX Server

Part 2

- Trade Messages
  - New Order
  - Execution Report
  - Order Cancel/Replace
- Trade Scenarios

Trade Messages

- New Order (Single)
- Execution Report
- Order Cancel Request
- Order Cancel/Replace Request
- Order Status Request etc

New Order (Single)

- Used to send a new (buy, sell etc) order to broker or an exchange.
- New order message provides numerous tags to support all possible information required with New order
- Has some mandatory fields that are common to every New order
  - Eg: ClientOrderID, Symbol etc
- Sample New Order Message:

8=FIX.4.1^9=0235^35=D^34=10^43=N^49=VENDOR^

50=CUSTOMER^56=BROKER^52=19980930-09:25:58^

1=XQCCFUND^11=10^21=1^55=EK^48=277461109^

22=1^54=1^38=10000^40=2^44=76.750000^10=165

New Order (Single)

- FIX Version (8) = 4.1
- Message Type (35) = D (New Order single)
- Message Seq (34) = 10
- PossDupFlag (43) = N (no)
- SenderCompID(49) = VENDOR (unique id of the sender firm)
- TargetCompID(56) = BROKER (value used to identify receiving firm)
- Checksum(10) –used for data integrity che

New Order (Single)

- Account(1) = Account number
- ClOrdID(11) = Client Order Id
- Symbol(55) = Security Identifier
- Side (54) = side of the order
- OrderQty(38) = Order Quantity
- OrdType(40) = Order Types
  - 1 = Market
  - 2 = Limit
  - 3 = Stop
  - 4 = Stop limit etc
- Price(44) = Price of order if the order is Limit etc

Execution Report

- Used for various needs like
  - Used to confirm the receipt of an order
  - Confirm changes to an existing order (in response to order cancel request etc)
  - Relay order status information
  - Reject orders
  - Relay Fill (execution) information etc

Execution Report

- Interpreted using ExecType, ExecTransType and OrdStatus fields
- ExecTransType

0 = NEW	Order Acknowledgement
1 = CANCEL	Cancel previously reported execution due to error etc. (Trade bust)
2 = CORRECT	Correction to the previously reported execution. (Trade restatements)
3 = STATUS	Reports the status STATUS of the orders.

Execution Report

- ExecType

0 = New	8 = Rejected
1 = Partial fill	A = Pending New
2 = Fill	C = Expired
4 = Canceled	E = Pending Replace (e.g. result of Order Cancel/Replace Request)
5 = Replace	6 = Pending Cancel (e.g. result of Order Cancel Request)

Execution Report

- OrdStatus

0 = New	8 = Rejected
1 = Partially filled	A = Pending New
2 = Filled	C = Expired
4 = Canceled	E = Pending Replace (e.g. result of Order Cancel/Replace Request)
5 = Replaced	6 = Pending Cancel (e.g. result of Order Cancel Request)

Execution Report

- OrderID(37) – Unique identifier for Order as assigned by broker
- CumQty(14) – Total number of shares filled
- LeavesQty(151) – Amount of shares open for further execution
- AvgPx(6) – Calculated average price of all fills on this order
- LastMkt(30) – Market of execution for last fill
- LastPx(31) – Price of this (last) fill
- LastShares(32) – Quantity of shares bought/sold on this (last) fill

ExecType Vs Order Status

- ExecType(150) – States the Execution Message type
  - New, Fill etc
  - Response to the request
- OrdStatus(39) – States the current orders status
  - New, Filled etc
  - May hold the same value as ExecType

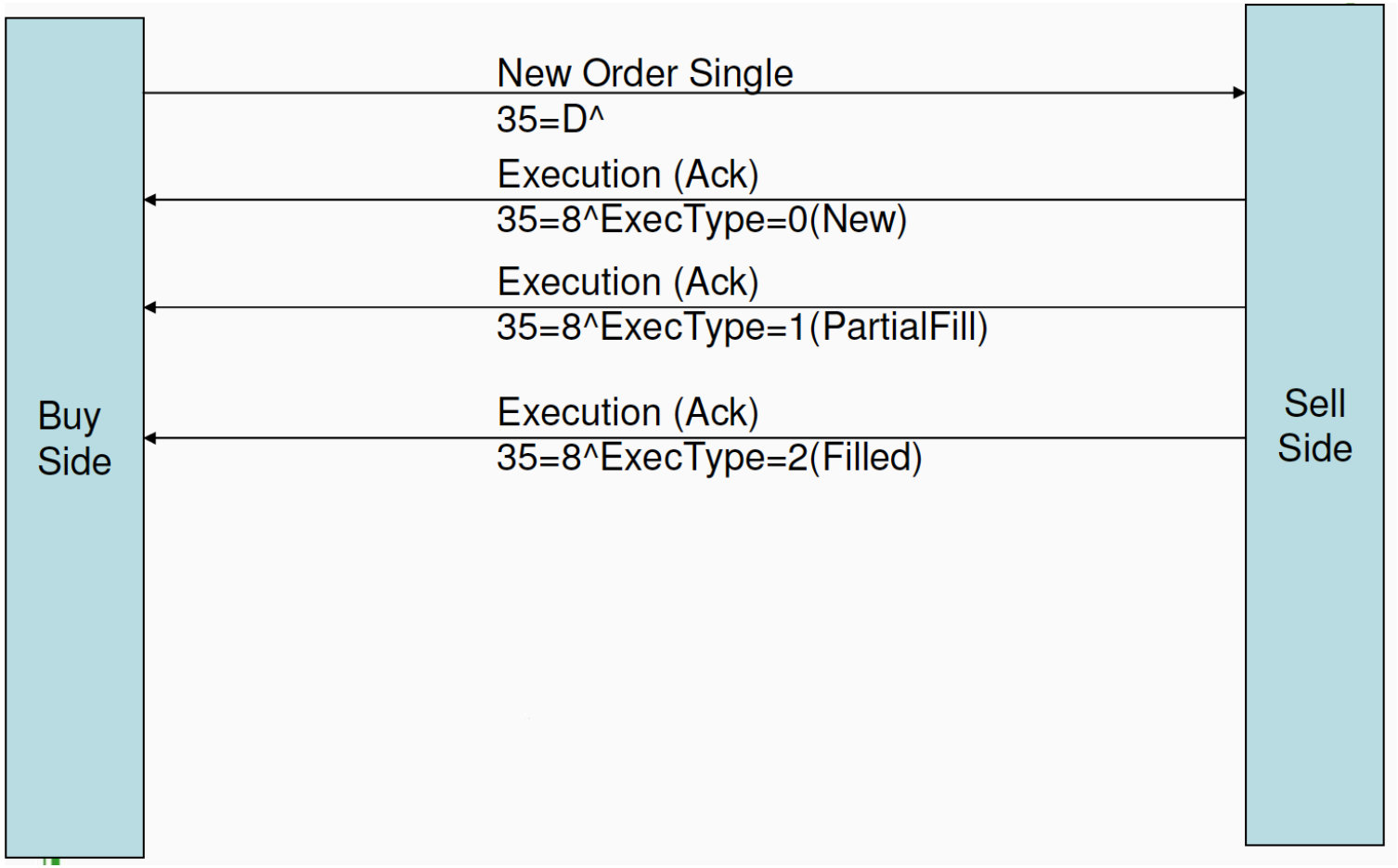
Order Cancel/Replace

- Also known as Order Modification
- Only modifiable properties can be changed
  - Order Qty, Order Price etc
  - Order Type: Limit -> Market
- Filled order can also be re-open by increasing the qty

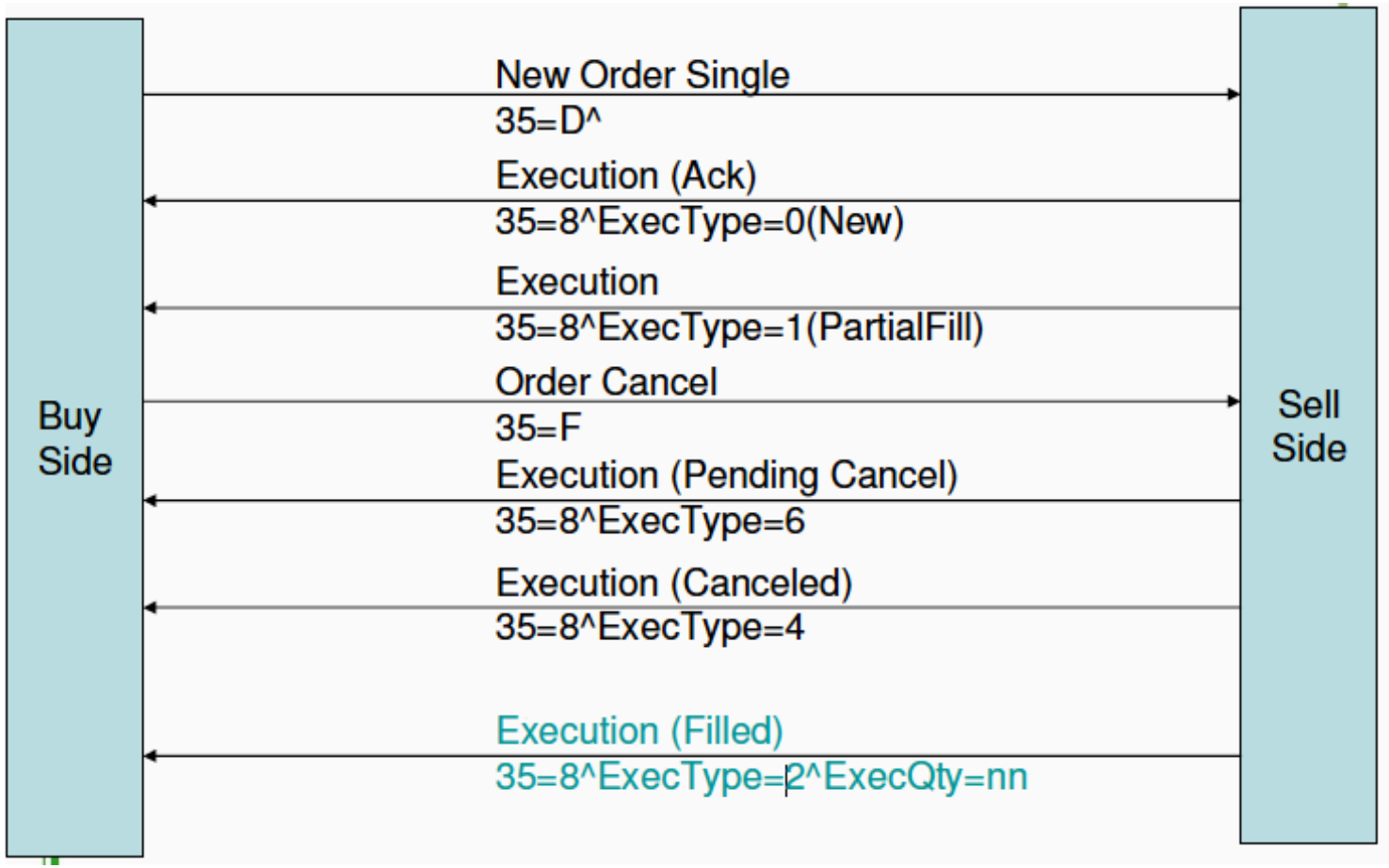
Few Other Application Messages

- Order Cancel Request
  - This message is used to request the cancellation of full or part of the remaining quantity of the existing order.
- Order Status Request Message
  - This message is used to request the status of existing(open) order

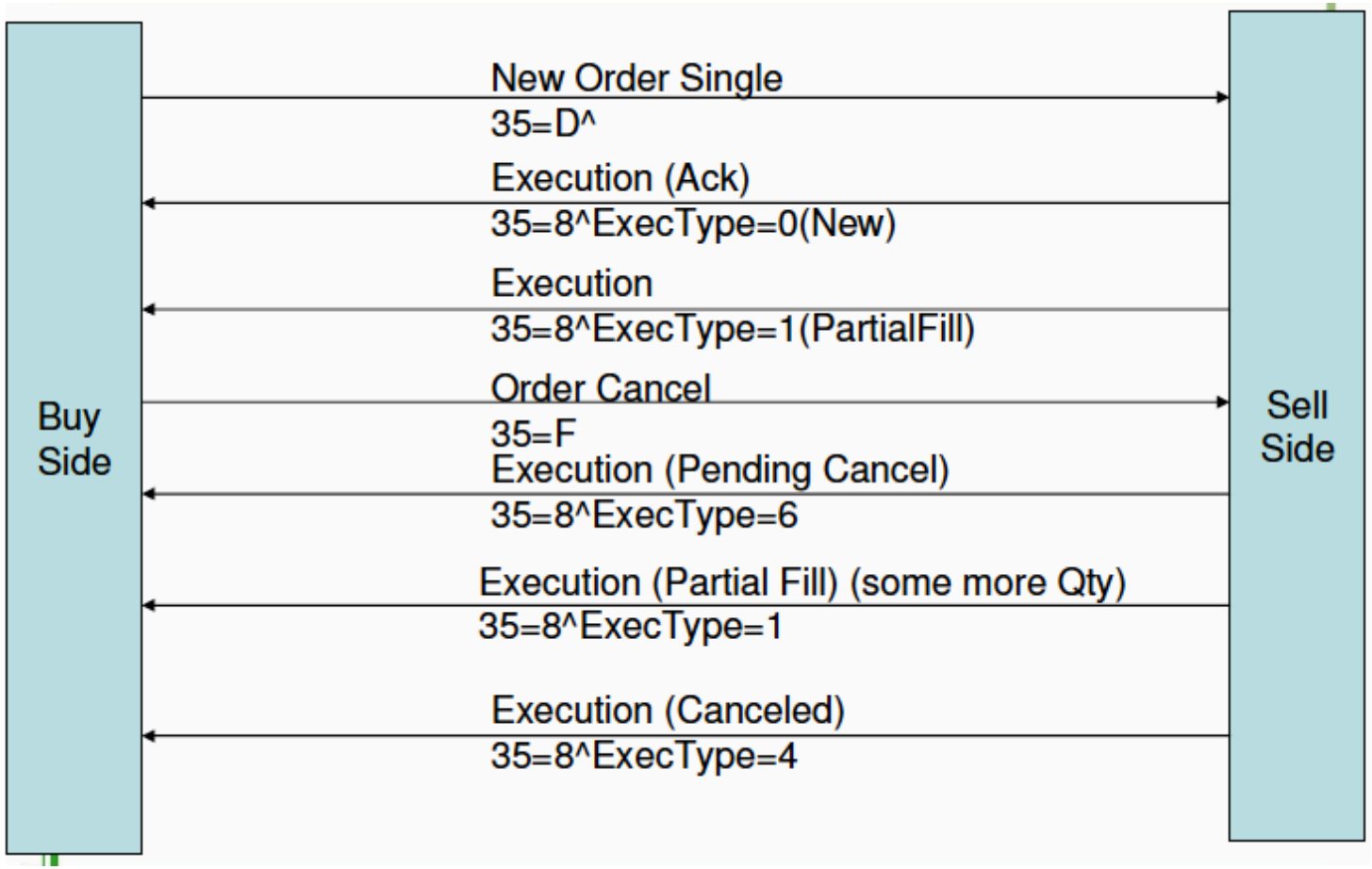
Scenario – 1 – Single Order



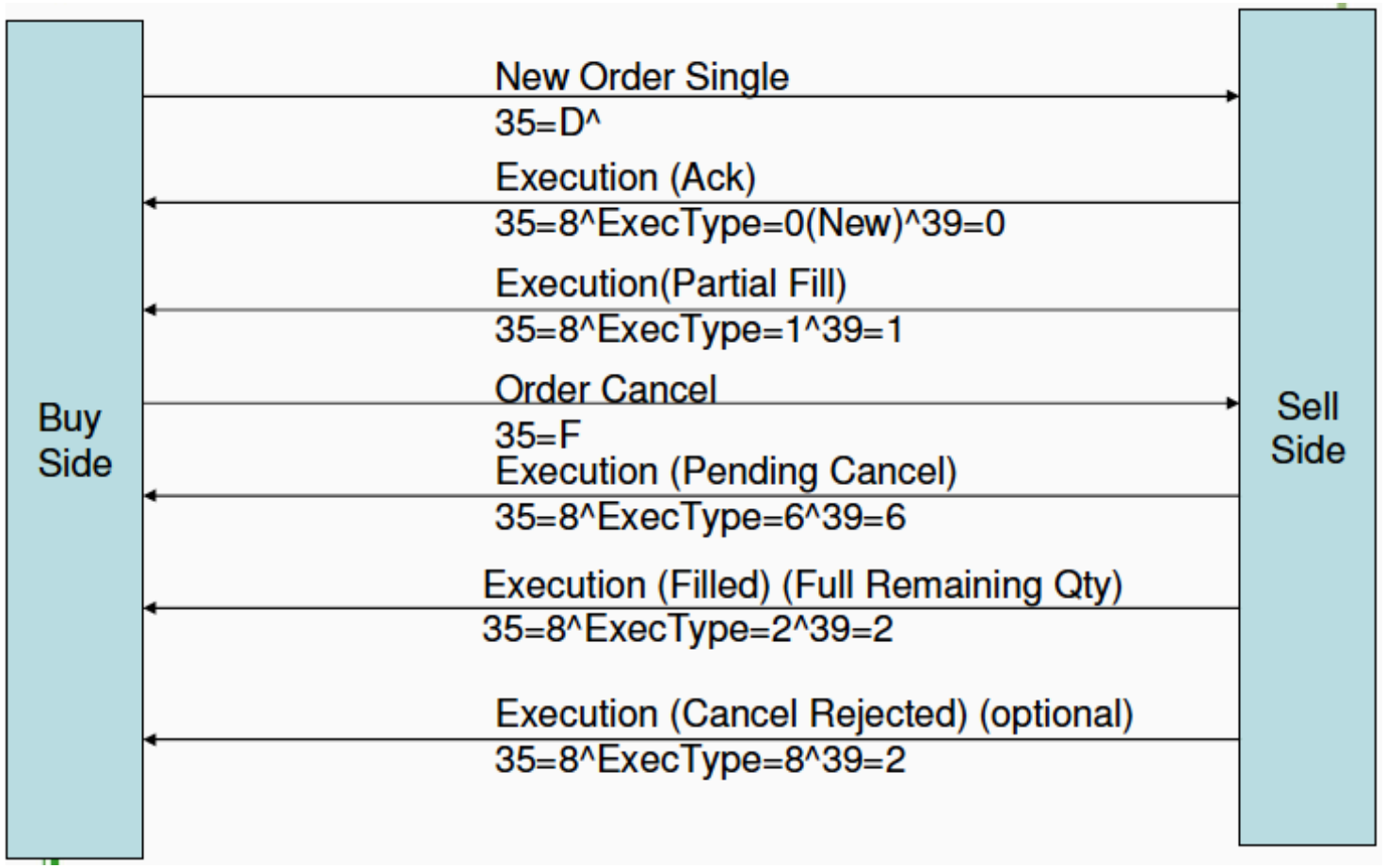
Scenario – 2 – Single Order



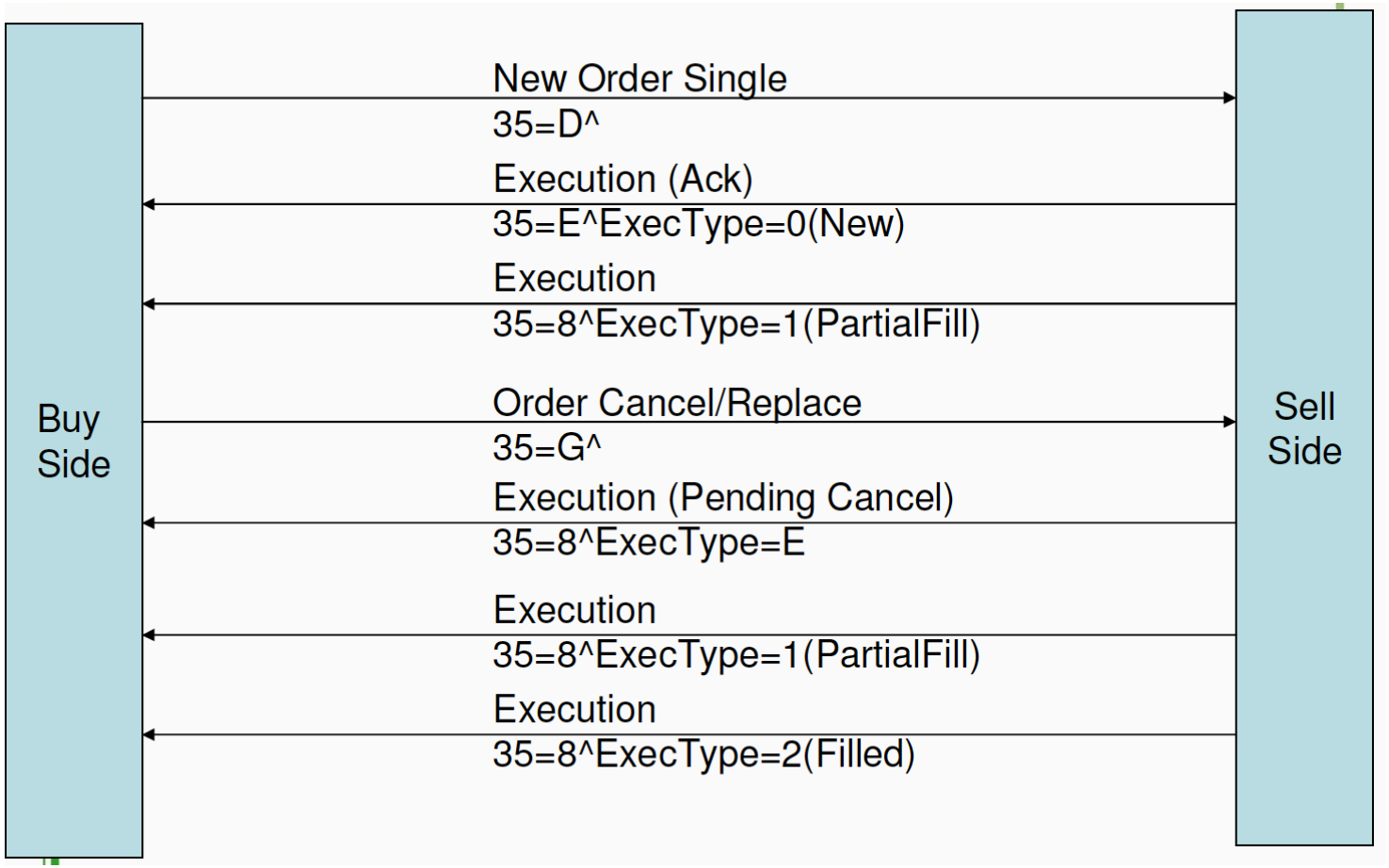
Scenario – 3 – Single Order



Scenario – 4 – Single Order



Scenario – 5 – Single Order



Part 3

- Technical Implementation (quickfixj)
  - Application
  - Session
  - Configurations
- Case study
  - Sequence number reset

Technical Implementation

- Based on quickfixj(open source FIX engine)
  - Application Interface
  - Session
  - Settings

Application Interface

- The primary interface for processing session messages

```
public interface Application {

    void onCreate(SessionID sessionId);

    void onLogon(SessionID sessionId);

    void onLogout(SessionID sessionId);

    void toAdmin(Message message, SessionID sessionId);

    void fromAdmin(Message message, SessionID sessionId)
            throws FieldNotFound, IncorrectDataFormat,
            IncorrectTagValue, RejectLogon;

    void toApp(Message message, SessionID sessionId)
            throws DoNotSend;

    void fromApp(Message message, SessionID sessionId)
            throws FieldNotFound, IncorrectDataFormat,
            IncorrectTagValue, UnsupportedMessageType;

    void fromApp(AcctMsg msg, SessionID sessionId);
}
```

Session Class

- The primary FIX abstraction for message communication
- Main functions
  - Send messages
  - Gap fill
  - Sequence number reset
  - Resend request
  - Logon
  - Logout
- Main methods
  - boolean sendToTarget(Message message)
  - *boolean sendAppMsgsToTarget(List<Message> messages)*

Settings

```
[default]
ConnectionType=initiator
HeartBtInt={HeartBtInt}
ReconnectInterval=3
JdbcLogHeartBeats=N

[session]
BeginString=FIX.4.2
SenderCompID={SenderCompID}
TargetCompID={TargetCompID}
SocketConnectHost={SocketConnectHost}
SocketConnectPort={SocketConnectPort}
NonStopSession=Y
StartTime=00:00:00
EndTime=00:00:00
TimeZone=UTC
ResetOnLogon={ResetOnLogon}
AllowUnknownMsgFields=Y
ValidateUserDefinedFields=N
DataDictionary={DataDictionary}
ValidateFieldsHaveValues=N
RejectInvalidMessage=Y
ValidateSequenceNumbers={ValidateSequenceNumbers}
```

Initiator

```
@Bean
public SocketInitiator socketInitiator(
        @Qualifier("application") Application application,
        MessageStoreFactory messageStoreFactory,
        SessionSettings settings,
        LogFactory logFactory) throws ConfigError {
    return new SocketInitiator(application, messageStoreFactory, settings,
            logFactory, new DefaultMessageFactory());
}
```

Case Study – Sequence Number Reset

- Reset if set tag ResetSeqNumFlag to Y in Logon messages
  - Reset session's ExpectedSenderNum and ExpectedTargetNum field to 1
- Two way to set ResetSeqNumFlag tag:
  - Set fix.resetOnLogon=Y in configurations
  - Set tag ResetSeqNumFlag = Y in toAdmin callback
- FIX session with IB, sequence number reset at every Saturday afternoon

Q&A

Thanks!

References

- fix_1day_allsections.pdf
- http://onixs.biz/fix-dictionary/4.2/
- http://www.quickfixj.org/