Basic Fix Connectivity Guide for ElectronX

Understanding FIX Protocol for Trading

Version: 1.0
Date: March 2026

WHO THIS GUIDE IS FOR

This guide is designed for:

- **Traders and operations staff** who need to understand what their technical teams are building
- **Developers new to financial markets** who are implementing FIX engines for the first time
- **Anyone asked to "connect via FIX API"** who doesn't know where to start
- **People who've hit roadblocks** trying to understand FIX documentation

What you'll learn:

- What FIX is and why exchanges use it (in plain English)
- The basic concepts you need to understand in order to successfully connect and trade
- How to think about FIX connections (not technical implementation details)
- Common problems and how to recognize, triage, and solve them
- When and how to ask for help

What this is NOT:

- A complete FIX protocol specification (see ElectronX FIX Specification)
- Programming tutorials (see QuickFix Setup Guide for implementation as well as online resources, such as from the Non-Profit FIX Protocol community)
- Business logic or trading strategy guidance. All materials provided are subject to the Participant and Clearing Member Agreement executed by all Participants and Clearing Members (including all disclaimers and limitations of liability contained therein) and the ElectronX (EXI) Rulebook.

Table of Contents

Change History Log

Version	Date	Description
1.0	29 January 2026	Initial document.

SECTION 1: WHAT IS FIX? (THE 5-MINUTE EXPLANATION)

The Simple Version

FIX stands for **F**inancial **I**nformation e**X**change. It's a standardized communications protocol that computerized trading/financial systems use to talk to each other. FIX is the lingua franca of financial markets.

What FIX Does:

- Enables you to send orders to an exchange (like ElectronX)
- Enables the exchange tell you when orders are filled, canceled, accepted, or rejected
- Enables you to receive market data (What orders to buy and sell are currently in the market? What are their prices and quantities? What trades are occurring? What symbols are available for trading?)
- Enables you to receive copies of all your trades for reconciliation from the exchange

What FIX Is:

- A **communications protocol** - messages are exchanged in a specific order
- A **structured message format** - (how to write your message) like filling out and exchanging forms with specific fields
- An **industry standard** - used by exchanges, brokers, and trading firms worldwide

What FIX Is NOT:

- A programming language (you use regular languages like C#, Python, Java to communicate with FIX)
- A proprietary system (it's a non-profit open standard anyone can implement)

A **FIX session** is like a phone call between your computer and ElectronX's servers.

Key characteristics:

- You "dial in" by sending a **Logon** message
- The session stays open as long as you're connected
- You "hang up" by sending a **Logout** message
- If the connection drops unexpectedly, you can reconnect and pick up where you left off

ElectronX has three types of sessions:

1. **Order Entry Session** - This is where you send orders (buy/sell) and receive confirmations
2. **Market Data Session** - This is where you subscribe to price updates and order book changes
3. **Drop Copy Session** - This is where you receive copies of ALL your firm's trades (for reconciliation)

Important: Each session is a separate connection with separate login credentials.

Why Exchanges Like ElectronX Use FIX

Standardization: FIX provides a common messaging framework. While each exchange defines its own implementation and ruleset, they are typically built on the same core FIX fields and concepts. Learning FIX for ElectronX therefore makes it much easier to adapt to other exchanges.

Automation: FIX allows for algorithmic trading - computers can place hundreds of orders per second.

Reliability: FIX is commonly implemented over Transmission Control Protocol (TCP), which provides built-in features for reliable transport, ensuring no orders or fill messages are lost even if you disconnect briefly.

SECTION 2: BASIC FIX CONCEPTS (THE BUILDING BLOCKS)

Concept 1: Messages (The Actual Conversation)

FIX is built on **messages** - structured pieces of information sent back and forth.

Think of messages like forms you fill out:

- Each message has a **type** (what kind of form is this?)
- Each message has **fields** (the boxes you fill in)
- Fields have **numbers** (field 55 is "Symbol", field 38 is "Order Quantity", etc.)

Example - Placing an Order:

When you want to buy 50 contracts of a product, you send a **NewOrderSingle** message that looks something like:

```
None
Message Type: D (NewOrderSingle)
Field 55 (Symbol): BFUT100-ERCOT-HB_NORTH-260122-15
Field 54 (Side): 1 (Buy)
Field 38 (Quantity): 50
Field 44 (Price): 45.00
Field 59 (Time In Force): 0 (Day order)
```

The exchange responds with an **ExecutionReport** message:

```
None
Message Type: 8 (ExecutionReport)
Field 39 (Order Status): 0 (New - order accepted)
Field 37 (Order ID): ABC123XYZ (the exchange's ID for your order)
Field 151 (Remaining Quantity): 50 (nothing filled yet)
```

You do not need to write these by hand! Your FIX engine (like QuickFix) handles the formatting. You just tell it "place this order" and it constructs the message.

Concept 3: Sequence Numbers (Keeping Track)

Every message has a **sequence number** - like page numbers in a book.

Why this matters:

- Your first message is #1, second is #2, third is #3, etc.
- The exchange's first message to you is #1, second is #2, etc.
- If you receive message #45 but the last one you saw was #43, you know you missed #44
- You can ask the exchange to **resend** missed messages

This is automatic - your FIX library handles this. But understanding it helps when troubleshooting.

Common problem: If you reset your sequence numbers by mistake but the exchange doesn't, you'll get disconnected. (The QuickFix Setup Guide explains how to avoid this.)

Concept 4: Sessions Are Weekly

ElectronX resets sequence numbers every Sunday at 06:15 UTC.

What this means:

- Every Sunday morning, both you and ElectronX agree to reset sequence numbers to 1
- Your messages from the previous week can't be replayed after the reset
- This is industry-standard practice

Why it matters: Your FIX configuration needs to account for this weekly reset. (Again, covered in the QuickFix Setup Guide.)

Concept 5: Heartbeats (Keeping the Connection Alive)

If no messages are sent for 30 seconds, FIX requires a **heartbeat** - a simple "I'm still here" ping.

Your FIX library handles this automatically.

Why you care: If heartbeats stop, the exchange assumes you've disconnected and closed the session. Common causes:

- Your client application crashed
- Network connection lost
- Firewall blocking traffic

SECTION 3: THE THREE TYPES OF SESSIONS (IN DETAIL)

Order Entry Session: Where You Trade

Purpose: Send orders (buy/sell) and receive execution confirmations.

What you can do:

- **Place orders** - NewOrderSingle message
- **Cancel orders** - OrderCancelRequest message
- **Modify orders** - OrderCancelReplaceRequest message

What you receive:

- **Order confirmations** - ExecutionReport (order accepted)
- **Fill notifications** - ExecutionReport (order executed, here's your trade)
- **Rejections** - ExecutionReport (order rejected, here's why)
- **Cancellation confirmations** - ExecutionReport (order canceled)

Key requirement: You must include a **SenderSubID** identifying the specific user/trader.

Example: SenderSubID=firms/ABC/users/trader1

This lets ElectronX know WHO at your firm is placing the order (required for audit trails).

Market Data Session: Where You Get Prices

Purpose: Receive real-time market data (prices, order book depth, trades).

What you can do:

- Request available contracts - SecurityListRequest (35=x)
- Subscribe to price updates - MarketDataRequest (35=V) for specific instruments
- Unsubscribe when done - Stop receiving updates for instruments
- Get reference data - Contract specifications, tick sizes, expiration dates

What you receive:

- **Initial snapshot** - Current state of the order book (all bids and offers)
- **Incremental updates** - Changes as they happen (new orders, trades, cancellations)
- **Trade notifications** - When trades occur in the market
- **Instrument list** - Active contracts available to trade

Two important notes:

1. You don't need SenderSubID for market data
2. Market data is "read-only" - you can't place orders through this session

Common use case: Your algorithm watches market data to decide when to place orders (which are then sent via the Order Entry session).

Drop Copy Session: Your Trade Record

Purpose: Receive copies of ALL executions across your firm for reconciliation and record-keeping.

What you receive:

- ExecutionReports for EVERY trade, regardless of which user/trader placed it

- Same format as Order Entry execution reports
- Includes a **trader identifier** (Tag 57) so you can tell who did what

Why it's useful:

- **Reconciliation** - Match your records against exchange records
- **Risk monitoring** - See a copy of all trading activity in real-time
- **Compliance/audit** - Comprehensive trade log

Key difference from Order Entry:

- This is **receive-only** - you can't send orders here
- Typically configured to show ALL firm activity, not just your own orders
- Used by risk managers, compliance, and operations teams

Example: Your firm has 5 traders. Each has their own Order Entry session. Your risk team connects to Drop Copy and sees trades from all 5 traders in one stream.

SECTION 4: DISCOVERING AVAILABLE CONTRACTS

The only way to programmatically discover available FIX symbols is through the **SecurityListRequest/SecurityList** messages.

Why this matters:

- ElectronX lists contracts on a rolling 5-day basis; new contracts are added daily
- You need exact symbol strings to place orders or subscribe to market data
- Symbols include specific date/hour encoding that must be precise

How to Get Available Contracts:

Send SecurityListRequest (35=x) on your Market Data session:

```
None
Message: SecurityListRequest (35=x)
Key fields:
- SecurityReqID (320): Your unique request ID
     - This is a unique identifier you generate to track this
specific request. Think of it like a confirmation number.
- SecurityListRequestType (559):
    * 4 = All Securities (recommended - get everything)
    * 0 = Individual symbol (if you know the exact symbol)
```

Receive SecurityList (35=y) response:

ElectronX responds with details for each available contract:

- **Symbol (55)** - The FIX symbol you need (e.g., BFUT500-ERCOT-HB_NORTH-260203-21)
- **MinPriceIncrement (969)** - Tick size (0.25 for all products)
- **MinTradeVol (562)** - Minimum quantity (usually 1)
- **SecurityGroup (1151)** - Product family grouping
- **Event dates (864/865/866)** - When contract starts, last trade date, expiration

Example Request:

```
None
Send: SecurityListRequest (35=x)
      SecurityReqID = "SEC_REQ_001"
      SecurityListRequestType = 4 (All Securities)

Receive: SecurityList (35=y)
         Symbol = BFUT100-ERCOT-HB_NORTH-260203-15
         Symbol = BFUT100-ERCOT-HB_NORTH-260203-16
         Symbol = BFUT500-ERCOT-HB_SOUTH-260203-15
         ... (120+ symbols for rolling 5-day window)
```

Best Practice: Cache and Refresh

1. **On connection:** Request full list (SecurityListRequestType=4)
2. **Parse and cache** all symbols locally
3. **Refresh daily** or when you need updated contracts
4. **Use cached symbols** for orders and market data subscriptions

Important: SecurityListRequest should be sent on your **Market Data session**, not Order Entry.

SECTION 5: A TYPICAL FIX WORKFLOW (STEP BY STEP)

Let's walk through what actually happens when you connect and trade:

Step 1: Connecting (The Logon)

```
None
YOU → EXCHANGE:  Logon message
                 "Hi, I'm ABC Trading, user trader1, let's start"

EXCHANGE → YOU:   Logon message
```

```
                "Hello ABC Trading, connection established"
[Session is now active]
```

Step 2: Discover Available Contracts

```
None
YOU → EXCHANGE:  SecurityListRequest (35=x)
                 "Give me all available contracts"
                 SecurityListRequestType = 4 (All Securities)

EXCHANGE → YOU:  SecurityList (35=y)
                 "Here are 120+ available contracts:
                  BFUT100-ERCOT-HB_NORTH-260203-15
                  BFUT100-ERCOT-HB_NORTH-260203-16
                  BFUT500-ERCOT-HB_SOUTH-260203-15
                  ... (full list with tick sizes and dates)"

Why this matters: You need these exact symbol strings to subscribe to
market data or place orders. The GUI doesn't show FIX symbol format.

Save these symbols: You'll use them in subsequent steps.
```

Step 3: Subscribing to Market Data (Optional but Common)

```
None
YOU → EXCHANGE:  MarketDataRequest
                    "Show me the order book for BFUT100-ERCOT-HB_NORTH-260122-15"

EXCHANGE → YOU:   MarketDataSnapshotFullRefresh
                    "Here's the current state:
                    Best bid 44.75 (50 lots),
                    Best offer 45.25 (100 lots)"

EXCHANGE → YOU:   MarketDataIncrementalRefresh (continuous)
                    "New bid entered: 45.00 (25 lots)"
```

```
                    "Trade occurred: 20 lots @ 45.00"
[Updates keep coming as market moves]
```

Step 4: Placing an Order

```
None
YOU → EXCHANGE:  NewOrderSingle
                  "Buy 50 lots @ 45.00, Day order"

EXCHANGE → YOU:  ExecutionReport
                   "Order accepted, Order ID = ABC123,
                   Status = New (resting in order book)"
```

Now you wait... Your order is sitting in the order book waiting for a match.

Step 5: Getting Filled

Scenario: Someone sells into (hits) your bid

```
None
EXCHANGE → YOU:   ExecutionReport
                  "Trade! Filled 30 lots @ 45.00
                   Cumulative filled = 30, Remaining = 20"

[A few seconds later]

EXCHANGE → YOU:   ExecutionReport
                  "Trade! Filled 20 lots @ 45.00,
                   Cumulative filled = 50, Remaining = 0
                   Order Status = Filled (complete)"
```

Your order is now fully executed. You bought 50 lots at 45.00.

Step 6: Canceling an Order (Alternative Scenario)

What if you wanted to cancel before getting filled?

```
None
YOU → EXCHANGE:  OrderCancelRequest
```

```
                  "Cancel Order ID ABC123"

EXCHANGE → YOU:   ExecutionReport
                  "Order Status = Canceled, Remaining = 20
                   (30 lots were filled before cancel took effect)"
```

Result: Your order is removed from the book. You got 30 lots filled before the cancellation was processed.

Step 7: Disconnecting (The Logout)

```
None
YOU → EXCHANGE:  Logout
                  "Goodbye, closing session"

EXCHANGE → YOU:   Logout
                  "Goodbye, session closed"
[Connection terminates]
```

Important: When you reconnect later, your sequence numbers continue from where they left off (unless it's Sunday morning when the week resets).

SECTION 6: GETTING STARTED - YOUR FIRST FIX CONNECTION

Overview: The Five-Stage Journey

Setting up FIX connectivity for the first time involves coordinating between multiple teams and ElectronX. Here's the roadmap:

```
None
Stage 1: Request Provisioning → Ask ElectronX to set up your gateways
Stage 2: Receive & Organize Materials → ElectronX sends you credentials and certificates
Stage 3: Configure Your Client → Set up your FIX library with the materials
Stage 4: Test & Validate → Prove your FIX sessions setup works in Beta,and meet the successful message counts for Production credentials (see the Production API Trading Approval Framework)
```

```
Stage 5: Receive Production Credentials and Set Up Your FIX Connection in the Production Environment
```

STAGE 1: REQUEST PROVISIONING FROM ELECTRONX

Before You Contact ElectronX - Quick Planning Checklist:

[ ] What sessions do you need?

- Order Entry (to place/cancel/modify orders) [ ]
- Market Data (to receive prices and order book updates) [ ]
- Drop Copy (to receive a record of all your trades for reconciliation) [ ]

[ ] How many of each session?

- Order Entry: One per trader or algorithm (for individual identification) [ ]
- Market Data: Usually one (shared across your systems), can request a second for backup [ ]
- Drop Copy: Usually one (covers all firm activity) [ ]

How to Request FIX Access:

Complete onboarding, in Phase 2 of the Onboarding process you will have the opportunity to indicate your FIX API trading preferences.

What Happens Next:

ElectronX will:

1. Confirm your onboarding status (must complete KYC onboarding first)
2. Provision your requested FIX gateways
3. Generate SSL certificates for your firm
4. Create a **FIX Connectivity Details** document customized for you
5. Send you credentials and access information

Timeline: Usually 2-3 business days for Beta provisioning

STAGE 2: RECEIVE & ORGANIZE YOUR FIX MATERIALS

What ElectronX Provides:

When your gateways are provisioned, you'll receive a package containing:

1. FIX Connectivity Details Document (PDF)

- Your specific SenderCompID(s)
- Gateway IP addresses and ports (Order Entry, Market Data, Drop Copy)
- Your SenderSubID format (for Order Entry)
- Session credentials

- Connection instructions

2. SSL Certificates (3 files)

- tls.key - Your private key (KEEP THIS SECRET!)
- cert.crt - Your certificate
- ca.pem - ElectronX's Certificate Authority certificate

3. Access to Documentation and Files

- ElectronX provides access to required documents and files via:
    - SFTP server (secure file transfer): recommended for automated retrieval
    - Shared folder (alternative for manual download)
- Available files include:
    - FIX Specification (message formats and fields)
    - FIX Reference Guide (session configuration details)
    - Data Dictionaries:
        - We provide .xml data dictionaries to define actual message structures and fields, XML stands for E**X**tensible **M**arkup **L**anguage which is structured, hierarchical human and machine readable language for defining electronic messages
        - FIX50SP2_OE.xml (application-layer FIX Order Entry .xml dictionary)
        - FIX50SP2_MD.xml (application-layer FIX Market Data .xml dictionary)
        - FIXT11_OE.xml (transport-layer FIX Order Entry .xml dictionary)
        - FIXT11_MD.xml (transport-layer FIX Order Entry .xml dictionary)
    - QuickFix Setup Guide
    - Basic FIX Connectivity Guide (this document)

Important Notes:

Beta vs. Production Separation:

- ***Beta and Production are completely separate systems***
    - Beta is a test environment, with fake/test collateral
    - Production is the genuine trading environment with bona fide collateral and trades
- Different SenderCompIDs
- Different hostnames and ports
- Different SSL certificates
- You'll receive Production credentials later (Stage 5) after completing Beta testing

Multiple Sessions:

- Each session type (Order Entry, Market Data, Drop Copy) has different credentials
- You may have multiple Order Entry sessions (one per trader/algorithm)
- Each requires separate configuration files (Stage 3)

STAGE 3: CONFIGURE YOUR FIX CLIENT

Now you're ready to configure your FIX client with the credentials and certificates provided by ElectronX.

Step 3.1: Choose and Install Your FIX Engine

ElectronX FIX gateways are compatible with any **FIX 5.0 SP2 compliant** implementation. **You must choose and install your own FIX engine.** ElectronX provides the FIX Specification and data dictionaries, but you are responsible for selecting the FIX engine. Examples include:

Open Source Libraries

- **Open Source Libraries:**
- **QuickFIX family** (C++, Java, C#/.NET, Python, Go) - Most commonly used
- **FixMaster** (.NET)
- **OnixS** (C++, Java, .NET) - Free community edition available

Commercial FIX Engines

- **Cameron FIX** (Multi-language)
- **B2BITS FIX Antenna** (C++, Java, .NET)
- **OnixS** (Full commercial version)
- **Marketcetera** (Java)

Build Your Own

- You can implement FIX from scratch using the FIX Protocol specification
- Only recommended if you have specific requirements not met by existing libraries

How to choose

- **Language compatibility:** Are there off-the-shelf engines available for your chosen language?
- **Timing:** How quickly do you need the FIX client online? What are lead times with commercial solutions?
- **Licensing:** Are you and your organization comfortable using open source software or willing to pay commercial access costs?
- **Support:** Are there active community forums for your chosen solution? Is targeted, paid support available for a commercial solution?
- **Features:** Built-in message replay, SSL support, monitoring tools?
- **Learning curve:** Available documentation and examples?

Why ElectronX provides QuickFix examples: Many ElectronX customers have chosen QuickFix, so we provide detailed configuration examples and troubleshooting guidance in the **QuickFix Setup Guide**. However, this is not an endorsement - you are free to use any FIX 5.0 SP2 compliant library.

What you'll need regardless of engine choice:

- ElectronX FIX Specification (for message formats and fields)
- ElectronX data dictionaries (FIX50SP2_OE.xml, FIX50SP2_MD.xml, etc.)
- Your SSL certificates (provided by ElectronX)
- Your FIX connectivity credentials (SenderCompID, host, port)

Once you've chosen and installed your FIX engine, proceed to configure it with ElectronX credentials.

Step 3.2: Create Your Configuration Files

Use the QuickFix Setup Guide (Section 4 - Complete Configuration Examples) as your template.

Create a separate config file for each session type:

```
None
config/beta/
├── order_entry_beta.cfg  # For adding/cxling orders
└── market_data_beta.cfg  # For receiving prices
```

Why separate files? Each session has different credentials and requirements. Keeping them separate makes management easier.

Step 3.3: Configure Order Entry Session [If Using Order Entry]

Start with the Order Entry example from the QuickFix Setup Guide, then customize these fields (using values from your FIX Connectivity Details document):

```
None
[DEFAULT]
# Replace with your desired file store path
FileStorePath=file/store/path
# Replace with your desired log file path
FileLogPath=file/log/path

ConnectionType=initiator
```

```
UseDataDictionary=Y
ValidateUserDefinedFields=Y
ValidateIncomingMessage=Y
TransportDataDictionary=PATH/TO/ORDER/ENTRY/GATEWAY/FIXT11.xml
AppDataDictionary=PATH/TO/ORDER/ENTRY/GATEWAY/FIX50SP2.xml

# Seqnum reset weekly on Sunday at 06:15 UTC (defined by EXI)
StartDay=Sunday
StartTime=06:15:00
EndDay=Sunday
EndTime=06:14:59
TimeZone=UTC

BeginString=FIXT.1.1
DefaultApplVerID=9

HeartBtInt=10
ReconnectInterval=20
ResetOnLogon=N
LogonTimeout=10

# Added debugging
ScreenLogShowIncoming=Y
ScreenLogShowOutgoing=Y
ScreenLogShowEvents=Y

TargetCompID=EXI
SenderSubID=YOUR_SENDER_SUB_ID
Instrument=YOUR_CHOSEN_INSTRUMENT

[SESSION]
SenderCompID=YOUR_SENDER_COMP_ID
# Replace with your account
Account=YOUR-ACCOUNT
```

```
CustomerOrderCapacity=2

[SESSION]
# Replace with your SenderCompID for drop copy session
SenderCompID=YOUR-SENDER-COMP-ID
```

Customizable Fields:

- SenderCompID - Can be found in your FIX Connectivity Details
- Certificate Paths - Paths are your operating system's addresses pointing to your credentials folder

Step 3.4: Configure Market Data Session [If Using Market Data]

Create market_data_beta.cfg:

```
None
[DEFAULT]
# Replace with your desired file store path
FileStorePath=YOUR/FILE/STORE/PATH
# Replace with your desired file store path
FileLogPath=YOUR/FILE/LOG/PATH

ConnectionType=initiator
UseDataDictionary=Y
ValidateUserDefinedFields=N
ValidateIncomingMessage=N
TransportDataDictionary=PATH/TO/MARKET/DATA/GATEWAY/FIXT11.xml
AppDataDictionary=PATH/TO/MARKET/DATA/GATEWAY/FIX50SP2.xml

# Seqnum reset weekly on Sunday at 06:15 UTC (defined by EXI)
StartDay=Sunday
StartTime=06:15:00
EndDay=Sunday
EndTime=06:14:59
TimeZone=UTC
```

```
BeginString=FIXT.1.1
DefaultApplVerID=9

HeartBtInt=10
ReconnectInterval=20
ResetOnLogon=N
LogonTimeout=10

ScreenLogShowIncoming=Y
ScreenLogShowOutgoing=Y
ScreenLogShowEvents=Y

SocketConnectPort=YOUR_SOCKET_CONNECT_PORT #Port Number
SocketConnectHost=YOUR.SOCKET.CONNECT.HOST #IP Address

TargetCompID=EXI
Instrument=YOUR-CHOSEN-INSTRUMENT
MarketDepth=5
SubscriptionType=1
[SESSION]

SenderCompID=YOUR-SENDER-COMP-ID
```

Key differences from Order Entry:

- Different SenderCompID (Market Data specific)
- **No SenderSubID** - not needed for Market Data
- Different host address and /port
- Uses **FIX50SP2_MD.xml** dictionary (Market Data specific message definitions)
- Uses **FIXT11_MD.xml** dictionary (Market Data specific message definitions)

Step 3.5: Understanding Critical Configuration Settings

These settings prevent common issues - DO NOT change them without understanding why:

```
None
# Sequence Number Management (Section 5 of QuickFix Setup Guide explains
in detail)
ResetOnLogon=N           # NEVER reset on normal logon
ResetOnLogout=N          # NEVER reset on logout
ResetOnDisconnect=N      # NEVER reset on abnormal disconnect
PersistMessages=Y        # ALWAYS save messages for replay

# Weekly Reset (ElectronX resets every Sunday 06:15 UTC)
Please note that this can be configured in multiple ways.
ResetOnTime=Y
ResetOnTimeDay=Sunday
ResetOnTimeTime=06:15:00
ResetOnTimeZone=UTC
```

What happens if these are wrong:

- ResetOnLogon=Y → You lose message history every time you reconnect
- ResetOnDisconnect=Y → Any network glitch will erase your message history
- PersistMessages=N → Cannot replay missed messages
- No ResetOnTime → Your sequence numbers diverge from ElectronX every Sunday

Step 3.6: Download ElectronX Data Dictionaries

Critical: You should use ElectronX's custom data dictionaries, not the default QuickFix ones, as we have customized our libraries to reflect our intended matching engine and market data gateway behaviors

Where to find them:

- SFTP / Shared Access Documents folder (provided by ElectronX)
- Your FIX materials package

Required files:

- FIX50SP2_OE.xml - Order Entry and Drop Copy application messages
- FIXT11_OE.xml - Order Entry transport layer
- FIX50SP2_MD.xml - Market Data application messages
- FIXT11_MD.xml - Market Data transport layer

Why ElectronX dictionaries are required:

- Include PossDupFlag support (fixes message replay issues)
- Include ElectronX-specific fields (ManualOrderIndicator, AggressorIndicator)
- Have correct enum values for rejection reasons

- Tested and validated against ElectronX gateways

See QuickFix Setup Guide Section 3 for detailed explanation.

Step 3.7: Coordinate with Your IT Team

Firewall Requirements:

Your IT team needs to open **outbound connections** to ElectronX gateways:

For Beta environment:

- Order Entry: 10.1.2.3:1 (use your actual host/port)
- Market Data: 10.1.2.3:2
- Drop Copy: 10.1.2.3:3

Protocol: TCP
Direction: Outbound only (you initiate connections, not inbound)
SSL/TLS: Yes (port is already SSL-enabled)

Common issue: Corporate firewalls often block outbound ports. Work with IT to whitelist ElectronX's specific hosts and ports.

How to verify firewall is open:

```
None
Work with your IT team to test connectivity using network tools.
Common approaches include:

- nc (netcat): `nc -zv <IP> <port>`
- openssl: `openssl s_client -connect <IP>:<port>`
- PowerShell (Windows): `Test-NetConnection -ComputerName <IP> -Port
<port>`

Replace `<IP>` and `<port>` with the values from your FIX Connectivity
Details document.

If the connection succeeds, the firewall is open. If connection times out
or is refused, the firewall is blocking traffic and IT will need to create
an exception.
```

Step 3.8: Create a Simple Logon/Logout Application

Utilizing your configured engine and network interfaces, create your own test application in a language of your choosing to verify if your connection works.

Minimal test app should:

1. Load your config file
2. Initiate connection
3. Log when Logon is received
4. Stay connected for 60 seconds (heartbeats will flow)
5. Send Logout and disconnect

Purpose: Proves configuration is correct before adding trading logic.

STAGE 4: TEST & VALIDATE

Now that your FIX client is configured, you must test all market data and order entry session behavior in Beta before requesting Production access. ***Note, all of the below is a strong suggestion; we encourage you to test your setup.***

Overview: Progressive Testing Approach

Do NOT try to test everything at once. Use this incremental approach:

```
None
Test 1: Connection → Can you Logon/Logout?
Test 2: Simple Order → Can you place and cancel?
Test 3: Order Fill → Can you execute trades?
Test 4: Market Data → Can you receive price updates?
Test 5: Reconnection → Can you handle disconnects?
Test 6: Week Rollover → Does Sunday reset work?
Test 7: Volume Testing → Meet Production message counts
```

Stop at each test if something fails. Fix issues before moving forward.

Test 1: Can You Connect? (10 minutes)

Goal: Verify that your client can establish a session.

Prerequisites:

- Configuration files created (Stage 3)
- SSL certificates in place
- Firewall opened by IT team

Steps:

1. Start your FIX client application with your Order Entry config file
2. Watch logs for outgoing Logon message
3. Watch for incoming Logon response from ElectronX
4. Verify session stays connected (heartbeats every 30 seconds)
5. Send Logout message

6. Verify clean disconnect

What to look for in logs:

```
None
# Successful connection initiation looks like:
[timestamp] Sent: 8=FIXT.1.1|9=...|35=A|... (Logon)
[timestamp] Received: 8=FIXT.1.1|9=...|35=A|... (Logon response)
[timestamp] Session established
[30 seconds later] Sent: 35=0 (Heartbeat)
[timestamp] Received: 35=0 (Heartbeat)
```

Success criteria:

- Logon sent (35=A outgoing)
- Logon received (35=A incoming)
- No error messages in logs
- Heartbeats flowing automatically
- Graceful logout (35=5)

Signs of Failure:

- **"Connection refused"** → Firewall blocking or wrong host/port (see QuickFix Setup Guide Section 7, Issue 3)
- **"SSL error"** → Certificate problem or path wrong
- **"Logon rejected"** → Wrong SenderCompID or credentials (verify against FIX Connectivity Details)

Do NOT proceed to Test 2 until you can consistently establish connection with the exchange.

Test 2: Can You Place an Order? (30 minutes)

Goal: Verify that your client can send a valid order.

Prerequisites:

- Test 1 passing consistently

Preparation:

1. Look up a valid contract symbol from the ElectronX FIX Specification
2. Check current market prices (in Beta GUI if available)
3. Choose a limit price away from market (order should NOT fill yet)

Steps:

1. Connect successfully (Test 1)
2. Send a NewOrderSingle (35=D):
    - Side: Buy (54=1)

    - Quantity: 1 contract (38=1)
    - Symbol: Valid contract from ElectronX
    - Order Type: Limit (40=2, only limit orders are currency supported on ElectronX)
    - Price: Away from market (44=[your price])
    - Time In Force: Day (59=0)
3. Receive ExecutionReport (35=8) with:
    - Order Status = New (39=0) → order accepted
    - Order ID provided (37=[exchange-assigned ID])
4. Cancel the order (35=F):
    - Include OrderID and OrigClOrdID
5. Receive ExecutionReport (35=8) with:
    - Order Status = Canceled (39=4)
6. Logout

What to look for in logs:

```
None
# Order placement:
Sent: 35=D|11=YOUR_CLORDID|55=SYMBOL|54=1|38=1|40=2|44=45.00|59=0|...
Received: 35=8|11=YOUR_CLORDID|37=EXCHANGE_ORDERID|39=0|... (Order accepted)

# Cancellation:
Sent: 35=F|41=YOUR_CLORDID|37=EXCHANGE_ORDERID|...
Received: 35=8|39=4|... (Order canceled)
```

Success criteria:

- Order accepted (39=0, not rejected)
- OrderID received from exchange (Tag 37)
- No field validation errors (Tag 58 would contain rejection reason)
- Cancel works (39=4)
- Order visible in ElectronX Beta GUI (if you have GUI access)

Signs of failure:

- **Order rejected (39=8)** → Check Tag 58 for reason
    - "Invalid symbol" → Verify symbol format from FIX Specification
    - "Invalid price increment" → Check tick size rules
    - "Invalid quantity" → Must be integer
    - "Insufficient collateral" → Check Beta account balance
- **"Invalid tag" errors** → Wrong data dictionary (must use FIX50SP2_OE.xml)
- **No response** → Check SenderSubID is included for Order Entry

Do NOT proceed to Test 3 until you can successfully place and cancel orders without error.

Test 3: Can You Get Filled? (1 hour)

Goal: Verify that your client can execute trades and receive fill confirmations.

Prerequisites:

- Test 2 passing (can place and cancel orders)
- Ensure selected product is actively quoted by other Beta accounts (ElectronX typically has trader bots quoting a number of markets in Beta for these purposes)

Steps:

1. Connect successfully (Test 1)
2. Check current market in Beta GUI:
    - Note best bid and best offer
3. Send an **aggressive order** that will fill immediately (Test 2):
    - To buy: Use price ABOVE current offer (with Side 54=1)
    - To sell: Use price BELOW current bid (with Side 54=2)
    - Quantity: 1 contract (38=1)
    - Time In Force: Day (59=0)
4. Receive ExecutionReport (35=8) with:
    - Order Status = Filled (39=2)
    - Execution Type = Trade (150=F)
    - LastPx = Fill price (31)
    - LastQty = Filled quantity (32)
    - CumQty = Total filled (14)
    - LeavesQty = 0 (151)
5. Verify trade appears in Beta GUI
6. Check your Beta account balance changed
7. Close position with opposite trade (if desired)
8. Logout

What to look for in Order Entry logs:

```
None
# Aggressive buy order:
Sent: 35=D|54=1|38=1|40=2|44=50.00|59=3|... (Buy 1 @ 50, IOC)

# Fill notification:
Received: 35=8|39=2|150=F|31=45.50|32=1|14=1|151=0|... (Filled 1 @ 45.50)
```

Success criteria:

- Order fills immediately (39=2)
- Receive ExecutionReport with trade details
- Trade appears in ElectronX Beta GUI
- Account balance reflects P&L
- Can parse fill details (LastPx, LastQty)

Understanding partial fills:

- If you try to buy 10 lots but only 3 are available:
    - First ExecutionReport: 39=1 (Partially Filled), CumQty=3, LeavesQty=7
    - Order remains in book waiting for more
    - Second ExecutionReport (when rest fills): 39=2 (Filled), CumQty=10, LeavesQty=0

Signs of failure:

- **Order rests in book (doesn't fill)** → Price not aggressive enough
- **Order rejected** → Check collateral, market status, symbol validity
- **Fill received but can't parse** → Check you're using ElectronX dictionaries

Test 4: Can You Handle Market Data? (1 hour)

Goal: Verify that your client can subscribe and receive price updates.

Prerequisites:

- Market Data config file created
- Market Data session credentials available

Steps:

1. Connect to Market Data session (separate from Order Entry!)
2. Send SecurityListRequest (35=x):
    - SecurityReqID (320) - Your Request Tracking ID:
    - SecurityListRequestType (559): 4 (All Securities)
3. Receive SecurityList (35=y):
    - Contains all available contracts with full FIX symbols
    - Parse and save at least one Symbol (55) value
    - Example symbol: BFUT100-ERCOT-HB_NORTH-260203-15
4. Send MarketDataRequest (35=V):
    - Subscription Type: Snapshot + Updates (263=1)
    - Market Depth: Full book (264=0)
    - Symbol: Choose active instrument
5. Receive MarketDataSnapshotFullRefresh (35=W):
    - Current order book snapshot

    - Multiple MDEntry groups (bids, offers)
6. Receive MarketDataIncrementalRefresh (35=X) messages:
    - Ongoing updates as market changes
    - New orders, trades, cancellations
7. Parse and display:
    - Best bid price and quantity
    - Best offer price and quantity
    - Recent trade prices
8. Unsubscribe:
    - Send MarketDataRequest with SubscriptionRequestType=2
9. Verify updates stop
10. Logout

What to look for in logs:

```
None
# Subscription:
Sent: 35=V|262=REQ_ID|263=1|264=0|146=1|55=SYMBOL|...

# Initial snapshot:
Received: 35=W|55=SYMBOL|268=10|269=0|270=44.75|271=50|... (10 MD entries)

# Incremental updates:
Received: 35=X|262=REQ_ID|268=1|279=0|269=0|270=44.50|271=25|... (New bid)
Received: 35=X|262=REQ_ID|268=1|279=2|278=ENTRY_ID|... (Delete)
```

Success criteria:

- Snapshot received with current order book
- Incremental updates arrive as market moves
- Can parse bid/offer prices (Tag 270 = MDEntryPx)
- Can parse quantities (Tag 271 = MDEntrySize)
- Updates match Beta GUI (if available)
- Unsubscribe works (updates stop)

Understanding MD message structure:

- MarketDataSnapshotFullRefresh: Full state of order book
- MarketDataIncrementalRefresh: Changes only (New, Change, Delete)
- Repeating groups: Tag 268 (NoMDEntries) indicates how many entries follow
- MDEntryType (Tag 269): 0=Bid, 1=Offer, 2=Trade

Signs of failure:

- **No snapshot received** → Check symbol is valid and actively traded
- **Can't parse entries** → Must use FIX50SP2_MD.xml dictionary (not OE dictionary!)
- **Connection rejected** → Verify Market Data credentials (different from Order Entry)

Test 5: Can You Reconnect? (30 minutes)

Goal: Verify that your client can handle disconnects gracefully

Prerequisites:

- Tests 1-3 passing

Steps:

1. Connect to Order Entry (Test 1)
2. Place an order that rests in the book but DO NOT cancel it (Test 2)
3. Note the MsgSeqNum of your last sent/received message
4. **Force disconnect** (close your application or enter ctrl+c to kill the process)
5. Wait 30 seconds
6. **Reconnect**
7. Your application should:
    - Send Logon with CURRENT sequence numbers (not reset to 1)
    - Continue from where it left off
8. If you missed messages (ElectronX sent while you were disconnected):
    - Send ResendRequest (35=2) for missed range (would need to log last sequence number before connection and compare to sequence number received when reconnected
    - Receive replayed messages with PossDupFlag=Y (Tag 43=Y)
9. Cancel the original order (it should still be in the book)
10. Receive cancellation confirmation
11. Logout

What to look for in logs:

```
None
# Before disconnect:
Last sent MsgSeqNum: 15
Last received MsgSeqNum: 28

# After reconnect:
Logon sent with MsgSeqNum=16 (continues from 15)
Logon received with MsgSeqNum=29 (continues from 28)
```

```
# If there's a gap (missed messages 26-28):
Sent: 35=2|7=26|16=28|... (ResendRequest for 26-28)
Received: 35=8|34=26|43=Y|... (Replayed ExecutionReport)
Received: 35=8|34=27|43=Y|... (Replayed ExecutionReport)
Received: 35=4|34=28|... (SequenceReset for admin messages)
```

Success Criteria:

- Reconnection works without manual intervention
- Sequence numbers continue (not reset to 1)
- ResendRequest returns missed messages
- Your application processes replayed messages (PossDupFlag=Y)
- Original order still in book and can be canceled

Signs of failure:

- **"MsgSeqNum too low"** → ResetOnLogon=N not set (see QuickFix Setup Guide Section 7, Issue 1)
- **ResendRequest doesn't work** → PossDupFlag dictionary issue (see QuickFix Setup Guide Section 5)
- **Sequence numbers reset to 1** → ResetOnDisconnect=N or PersistMessages=Y not set
- **Replayed messages dropped** → Using wrong dictionary or not handling PossDupFlag

This test is CRITICAL. Production systems must handle disconnects gracefully.

Test 6: Can You Handle a Week Rollover? (Ongoing)

Goal: Verify that your client can handle the weekly reset correctly.

Background: ElectronX resets all FIX sessions every Sunday at 06:15 UTC. Sequence numbers reset to 1 for all sessions.

When to test:

- Wait for Sunday 06:15 UTC rollover
- OR: ElectronX support can manually trigger a reset in Beta for testing

Steps:

1. Connect Friday and place some orders
2. Stay connected through Sunday 06:15 UTC (or disconnect and reconnect after)
3. At 06:15 UTC, your QuickFix should:
    - Send Logon with ResetSeqNumFlag=Y (Tag 141=Y)
    - Reset sequence numbers to 1
4. Continue trading normally after reset
5. Verify no "MsgSeqNum too low" errors

What to look for in logs:

```
None
# Before reset (Friday):
MsgSeqNum at 127

# At Sunday 06:15 UTC:
Sent: 35=A|34=1|141=Y|... (Logon with ResetSeqNumFlag)
Received: 35=A|34=1|141=Y|... (Logon response with ResetSeqNumFlag)

# After reset:
MsgSeqNum back to 1, continues normally
```

Success criteria:

- No "MsgSeqNum too low" errors after Sunday reset
- Sequence numbers properly reset to 1
- Trading continues normally in new week
- Can still replay messages from current week (but not previous week)

Signs of failure:

- **"MsgSeqNum too low" after Sunday** → ResetOnTime settings wrong
- **No automatic reset** → Check ResetOnTime=Y, ResetOnTimeDay=Sunday, etc.
- **Timezone wrong** → Must use UTC (ResetOnTimeZone=UTC)

Test 7: Volume Testing - Meeting Production Message Counts (Days to Weeks)

Goal: Demonstrate FIX proficiency by meeting minimum message counts required for Production approval.

Background: ElectronX requires proof of competency before granting Production access. See the **Production API Trading Approval Framework** for complete requirements.

Minimum message counts in Beta:

- 50+ NewOrderSingle (35=D) - Orders placed
- 25+ OrderCancelRequest (35=F) - Cancellations
- 10+ OrderCancelReplaceRequest (35=G) - Modifications
- Multiple successful ResendRequest (35=2) - Message replay
- Market Data subscription and incremental data handling
- At least one weekly session reset successfully handled

Additional requirements:

- No repeated configuration errors in past 2 weeks
- Handle rejections gracefully (don't flood exchange with bad orders or messages)

- Demonstrate understanding of order lifecycle (New → Partial Fill → Fill)
- Successfully handle IOC, FOK, and DAY orders

Suggested testing timeline:

Week 1: Basic Operations

- Days 1-2: Tests 1-3 (connection, simple orders, fills)
- Days 3-4: Test 4 (market data)
- Day 5: Test 5 (reconnection)

Week 2: Volume and Variety

- Build up order counts (10-15 orders per day)
- Test different order types (IOC, FOK, Day)
- Test order modifications (cancel-replace)
- Practice reconnection multiple times

Week 3: Production Readiness

- Continue trading to reach 50+ orders
- Ensure all message types tested
- Sunday rollover testing (Test 6)
- Final validation and preparation

How to track your progress:

- Review your FIX logs to count message types
- Use ElectronX Beta GUI to see your trading history
- Keep notes of issues encountered and resolved

Success criteria for Production approval:

- All minimum message counts achieved
- All test scenarios passed
- No configuration issues in past 2 weeks
- Demonstrated understanding of FIX protocol
- Ready to trade with real money

Test Summary Checklist:

[ ] Test 1: Connection - Logon/Logout working

[ ] Test 2: Simple Orders - Place and cancel

[ ] Rest 3: Order Fills - Execute trades

[ ] Test 4: Market Data - Receive and parse updates

[ ] Test 5: Reconnection - Handle disconnects and replay

[ ] Test 6: Week Rollover - Sunday reset handling

[ ] Test 7: Volume Testing - Meet Production message counts

When all tests pass: You're ready to request Production access

STAGE 5: RECEIVE PRODUCTION CREDENTIALS AND SET UP YOUR FIX CONNECTION IN PRODUCTION

Prerequisites for Production Access:

Before ElectronX will provision Production gateways, you must:

- **Complete Beta Testing Requirements**
- **Account funding:**
    - Minimum funding requirement of $25,000

Requesting Production Access:

Once you've met all Beta testing requirements, you can email support@electronx.com to request Production access. We will check your successful message counts, let you know if anything is missing, and provide you with Production credentials.

ElectronX Production Review Process:

1. **EXI team will review your Beta activity**
    - Examine your Beta logs for message patterns
    - Verify minimum message counts achieved
    - Check for repeated configuration errors or issues
    - Assess overall technical proficiency

2. **EXI team will verify account status**
    - Confirm onboarding complete
    - Verify collateral deposited and cleared
    - Check no outstanding compliance issues

3. **EXI teams will give internal approve or deny access**
    - Requires Market Operations approval
    - Requires Risk Management approval
    - Compliance approval (if applicable)

4. **EXI team will provision production gateways**
    - Create Production CompIDs (different from Beta)
    - Generate Production SSL certificates (different from Beta)
    - Configure gateway access controls

5. **Send Production Materials**

- New FIX Connectivity Details document (Production)
- New SSL certificates (Production)
- Go-live coordination instructions

Timeline: Allow 1-2 weeks for Production provisioning and approval process

Receiving Your Production Materials:

ElectronX will send a **new** FIX Connectivity Details document for Production:

CRITICAL: Production credentials are completely separate from Beta. You cannot use Beta certificates or CompIDs in Production.

STAGE 5 CHECKLIST

[ ] PRODUCTION READINESS VALIDATION

- Met all message counts in Beta (see Production API Trading Approval Framework) [ ]
- Collateral deposited and cleared [ ]
- Internal approvals obtained (trading, risk, compliance) [ ]

[ ] PRODUCTION PROVISIONING

- Sent Production provisioning request to support@electronx.com [ ]
- Received Production FIX Connectivity Details document [ ]

[ ] PRODUCTION CONFIGURATION

- Set up account in Production [ ]

SECTION 6A: COMMON FIRST-TIME SETUP PROBLEMS

Problem 1: "I Can't Download the Certificates"

Symptoms: ElectronX sent a link but you can't access the files

Causes:

- Corporate network blocking ElectronX's file sharing
- Email security stripping attachments
- Incorrect link format

Solutions:

- Ask ElectronX to use alternative delivery (SFTP, secure portal)
- Use personal email temporarily to download, then transfer internally
- Contact your IT security team to whitelist ElectronX domains

Problem 2: "Connection Refused"

Symptoms: Can't connect at all, timeout or "connection refused" errors

Causes:

- **Firewall blocking** (most common)
- Wrong host or port in config
- Wrong/missing certificates (prod certificates used against beta, etc)
- Certificates not locatable (location not set properly in stunnel cfg)
- Gateway not provisioned yet/gateway down

Solutions:

1. **Verify with ElectronX:** "Can you confirm my gateway is provisioned and my SenderCompID is correct?"
2. **Test from command line:**

```shell
telnet beta-oe.electronx.com 10001 # If this doesn't respond with "Connected to .....", then there is a network/firewall issue
```

3. **Work with IT:** Show them the exact host/port from your FIX Connectivity Details
4. **Try STunnel:** If direct SSL isn't working, STunnel can help isolate the problem

Problem 3: "Logon Rejected"

Symptoms: Connection succeeds but Logon is rejected

Causes:

- Wrong SenderCompID in config
- Using Production credentials on Beta (or vice versa)
- Using the wrong IP, wrong port, wrong TargetCompID, etc.
- Gateway not yet provisioned for your CompID

Solutions:

1. **Double-check credentials:** Compare your config against your FIX Connectivity Details letter-by-letter
2. **Check environment:** Are you connecting to Beta with Beta credentials (not Production)?
3. **Contact ElectronX:** "I'm getting Logon rejected with SenderCompID=ABCOE01, can you verify this is provisioned?"

Problem 4: "Orders Immediately Rejected"

Symptoms: Logon works, but every order is rejected

Causes:

- Invalid symbol format
- Invalid price (wrong tick size)
- Invalid quantity (must be integer)

- Market closed
- Insufficient collateral
- Wrong SenderSubID format (for Order Entry)

Solutions:

1. **Check rejection text:** Tag 58 in ExecutionReport is a text string that describes WHY
2. **Compare to working example:** Place an order in ElectronX GUI, then try to replicate via FIX
3. **Verify symbol:** Copy exact symbol from FIX Specification or ElectronX GUI
4. **Check market status:** Is the trading session currently open?

Problem 5: "Everything Worked Yesterday, Now Nothing Works"

Symptoms: Was working fine, suddenly broke

Causes:

- **Weekly reset happened** (Sunday 06:15 UTC) and config doesn't handle it
- **Beta environment reset** (ElectronX occasionally resets Beta)
- **Ran out of demo funds** (Beta has limited artificial funds)
- **Certificate expired** (rare, but possible)
- **Market schedule change** (holiday, different hours)

Solutions:

1. **Check if today is Sunday/Monday after reset**
2. **Check account balance** in ElectronX GUI
3. **Contact ElectronX:** "Was there any maintenance or reset on [date]?"
4. **Compare logs:** What was the last successful message before it broke?

SECTION 6B: SETUP CHECKLIST

Use this checklist to track your progress:

[ ] STAGE 1: PREREQUISITES

- Defined use case (algo trading, manual assistance, etc.) [ ]
- Identified team members (developer, IT, trader) [ ]
- Determined which sessions needed (Order Entry, Market Data, Drop Copy) [ ]
- Decided how many connections [ ]
- Choose programming language/FIX library [ ]

[ ] STAGE 2: PROVISIONING REQUEST

- Completed ElectronX onboarding [ ]
- Sent provisioning request email to support@electronx.com [ ]
- Received confirmation from ElectronX [ ]
- Received FIX Connectivity Details document [ ]

[ ] STAGE 3: MATERIALS RECEIVED

- Downloaded SSL certificates (client.key, client.crt, ca.crt) [ ]
- Downloaded FIX Specification [ ]
- Downloaded FIX Reference Guide [ ]
- Downloaded Data Dictionaries (FIX50SP2_OE.xml, etc.) [ ]
- Organized files into folder structure [ ]

[ ] STAGE 4: CONFIGURATION

- Created configuration file from template [ ]
- Customized with your credentials (SenderCompID, host, port) [ ]
- Configured SSL certificates [ ]
- Added ElectronX data dictionaries [ ]
- Set critical settings (ResetOnLogon=N, PersistMessages=Y, etc.) [ ]
- Coordinated with IT on firewall rules [ ]

[ ] STAGE 5: TESTING

- Test 1: Connection and Logon successful [ ]
- Test 2: Can place and cancel orders [ ]
- Test 3: Can get filled [ ]
- Test 4: Can receive market data [ ]
- Test 5: Can reconnect and replay messages [ ]
- Test 6: Handled weekly reset (if occurred) [ ]
- Reviewed Production API Trading Approval Framework [ ]

SECTION 7: COMMON SCENARIOS EXPLAINED

Scenario A: "My Order Was Rejected"

What happened: You sent a NewOrderSingle, but instead of "Order accepted," you got an ExecutionReport with Status = Rejected.

Common reasons:

1. **Invalid price increment**
    - Example: You sent 45.13 but ElectronX only allows 0.25 increments (45.00, 45.25, 45.50, 45.75)
    - Fix: Check the FIX Specification for tick sizes

2. **Invalid quantity**
    - Example: You sent 10.5 contracts but quantities must be whole numbers
    - Fix: Ensure order quantity is an integer

3. **Insufficient collateral**

    - ElectronX checks pre-trade risk. If you don't have enough funds, order is rejected
    - Fix: Add funds to your account

4. **Unknown symbol**
    - Example: Typo in the contract symbol
    - Fix: Double-check symbol formatting (see FIX Specification)

5. **Market is closed or halted**
    - Can't trade when the market isn't open
    - Fix: Wait for market to open

The **rejection** message **includes** a text field **(Tag 58) explaining WHY.** Always read this field!

Scenario B: "I Got Partially Filled"

What happened: You placed an order for 100 lots, but only 30 executed. Your order is Status = Partially Filled.

Why this happens:

- Not enough liquidity on the other side
- Your limit price wasn't aggressive enough
- The 30 lots that filled were all that was available at your price

What happens next:

- Your order remains in the book for the remaining 70 lots
- If more counter-side orders arrive at your price, you'll get additional fills
- If you used Time In Force = IOC (Immediate or Cancel), the remaining 70 would be automatically canceled
- If you used Time In Force = FOK (Fill or Kill), the ENTIRE order would be rejected if it couldn't fully execute immediately

Your options:

1. **Wait** - Let the order rest and hope for more fills
2. **Cancel** - Cancel the remaining 70 lots
3. **Modify** - Send an OrderCancelReplaceRequest to change price or quantity

Scenario C: "I Disconnected, What Happens to My Orders?"

It depends on your Cancel on Disconnect (CoD) settings:

CoD Enabled (ElectronX default):

- All **DAY orders** are automatically canceled when you disconnect
- **GTC orders** (Good 'Til Canceled) remain in the book

- When you reconnect, you can request replays of messages you missed (including the cancel confirmations)

CoD Disabled:

- All orders remain in the book even when disconnected
- Your orders keep getting filled even though you're not connected
- Generally not recommended

Best practice: Keep CoD enabled for DAY orders. Use GTC only if you truly want orders to persist across disconnects.

Scenario D: "I Missed Some Messages, How Do I Get Them Back?"

The ResendRequest:

If you realize you missed messages (because sequence numbers jumped), you send a ResendRequest:

```
None
YOU → EXCHANGE:  ResendRequest
                  "Please resend messages 46 through 50"

EXCHANGE → YOU:   [Original messages with PossDupFlag=Y]
                  "Here's message 46 (marked as possible
duplicate)"
                  "Here's message 47 (marked as possible
duplicate)"... etc
```

PossDupFlag=Y means "this is a replayed message, you may have already seen it."

Common issue: Some FIX configurations don't handle PossDupFlag correctly and drop replayed messages. The QuickFix Setup Guide explains how to fix this.

SECTION 8: TROUBLESHOOTING MINDSET

When things go wrong, use this mental framework:

The Three Layers of FIX Problems

Layer 1: Network/Connection (Can you connect at all?)

- Symptoms: "Connection refused," "timeout," "can't reach server"
- Common causes: Wrong host/port, firewall blocking, SSL certificate issues
- Who helps: Your IT team, ElectronX support for host/port verification

Layer 2: Session Management (Can you logon and stay connected?)

- Symptoms: "MsgSeqNum too low," disconnects every 30 seconds, "Logon rejected"

- Common causes: Sequence number misconfiguration, heartbeat settings, credentials
- Who helps: QuickFix configuration expertise, FIX Protocol community forums, paid commercial support (if applicable), ElectronX support

Layer 3: Application Logic (Can you place orders and trade?)

- Symptoms: Orders rejected, unexpected fills, missing fields
- Common causes: Message formatting, field values, business rules
- Who helps: Your developers, ElectronX FIX Specification, ElectronX support

Reading FIX Logs

Your FIX library writes logs showing every message sent and received. Learn to read them!

Example log snippet:

```
None
20260122-14:35:12.123 : Sent: 8=FIXT.1.1|9=156|35=D|34=5|49=ABC|...
20260122-14:35:12.156 : Received: 8=FIXT.1.1|9=245|35=8|34=12|49=EXI|39=0|...
```

What this tells you:

- 35=D means you sent a NewOrderSingle
- 35=8 means you received an ExecutionReport
- 39=0 means Order Status = New (accepted)

Learn the important message types:

- 35=A - Logon
- 35=5 - Logout
- 35=D - NewOrderSingle (place order)
- 35=8 - ExecutionReport (order update)
- 35=F - OrderCancelRequest (cancel order)
- 35=G - OrderCancelReplaceRequest (modify order)
- 35=V - MarketDataRequest (subscribe)
- 35=W - MarketDataSnapshotFullRefresh (initial data)
- 35=X - MarketDataIncrementalRefresh (updates)

The "It Worked Yesterday" Problem

If something suddenly stops working:

1. **Check if the weekly reset happened**
    - Did you reconnect on Sunday? Session might have reset
2. **Check the market schedule**
    - Are you trying to trade when the market is closed?

3. **Check your collateral**
    - Did you run out of available funds?
4. **Check for ElectronX system messages**
    - Did ElectronX send an email about maintenance or changes?
5. **Compare logs from "when it worked" vs. "now"**
    - What's different in the messages?

SECTION 9: WHEN TO ASK FOR HELP (AND HOW)

Self-Service First:

You can check these before contacting support:

- ElectronX FIX Specification (for message formats and field definitions)
- QuickFix Setup Guide (for configuration issues)
- This guide (for conceptual understanding)
- Your FIX logs (what do the actual messages say?)

When to Contact ElectronX Support:

DO contact support for:

- ElectronX FIX gateway behavior questions
- Message interpretation (what does this rejection mean?)
- Connectivity details (host, port, credentials)
- Collateral/account balance questions
- Market data feed issues
- Suspected exchange-side bugs

How to Ask for Help Effectively:

Bad request:

> "My FIX connection doesn't work, help!"

Good request:

> "I'm getting 'MsgSeqNum too low' rejections when connecting to Order Entry. I've attached my config file (credentials redacted) and the last 50 lines of my FIX log. This started happening after reconnecting this morning. My sequence numbers are at 127 but ElectronX expects 1."

Why is this a good request:

- Specific error message
- Relevant logs attached
- Context (when did it start?)
- What you've observed (sequence number mismatch)

- Which session (Order Entry vs Market Data vs Drop Copy)

Email support with:

1. Your ElectronX account/firm name
2. Which session type (Order Entry, Market Data, Drop Copy)
3. Specific error messages or symptoms
4. Your FIX logs (last 30-60 minutes)
5. Your configuration file **(REDACT your credentials!)**
6. Explanation of what you were trying to do and what failure occurred
7. What you've already done to debug

SECTION 10: KEY TAKEAWAYS

Remember these core principles:

1. **FIX is a conversation** - Messages go back and forth between you and ElectronX
2. **Sessions must be maintained** - Logon, heartbeat, manage sequence numbers
3. **Order Entry, Market Data, and Drop Copy are separate sessions** - Each with its own purpose
4. **Messages have types and fields** - Learn to recognize the important ones
5. **Your FIX engine handles most complexity** - You configure it, then focus on business logic
6. **Logs are your friend** - Learn to read them for troubleshooting
7. **Test in Beta first** - Don't learn on live money

The path to success:

1. Read this guide to understand concepts
2. Read the FIX Specification to understand message details
3. Read the QuickFix Setup Guide to configure your connection
4. Test thoroughly in Beta
5. Ask for help when stuck (but try to help yourself first)
6. Go to production with confidence

APPENDIX A: GLOSSARY OF TERMS

Term	Definition
FIX	**F**inancial **I**nformation e**X**change protocol - the language exchanges and trading systems use to communicate

Session	A persistent, stateful endpoint, provisioned by the exchange, that maintains message sequence numbers and through which a customer sends and receives FIX messages.
Message	A structured collection of information/tags exchanged via FIX
Tag	A numbered field in a FIX message (e.g., Tag 55 = Symbol, Tag 38 = OrderQty)
Sequence Number	A number (Tag 34) that keeps messages in order within a session. Each side maintains and numbers its own outbound messages independently starting at 1.
Logon	The message (35=A) sent by both sides to start a FIX session
Logout	The message (35=5) sent by either side to end a FIX session
Heartbeat	A keep-alive message (35=0) sent after 30 seconds of inactivity to maintain the session.
ExecutionReport	A message (35=8) from the exchange about an order (accepted, filled, rejected, canceled)
NewOrderSingle	The message (35=D) you send to place an order
OrderCancelRequest	The message (35=F) you send to cancel an order
OrderCancelReplaceRequest	The message (35=G) you send to modify an order's price and/or quantity
Order Entry	The session type for receiving read-only copies of all order activity from your order entry session
Market Data	The session type for receiving price and order book information

Drop Copy	The session type for receiving copies of all order entry session traffic
ClOrdID	Client Order ID (Tag 11) - your unique identifier for each order
OrderID	Exchange-assigned Order ID (Tag 37) - ElectronX's identifier for your order
OrigClOrdID	Original Client Order ID (Tag 41) - references which order to cancel or modify
OrdStatus	Order Status (Tag 39) - shows order state: New, Partially Filled, Filled, Canceled, Rejected, Expired
ExecType	Execution Type (Tag 150) - what happened to the order: New, Fill, Partial Fill, Canceled, Replaced, Rejected
Side	Buy (1) or Sell (2) - Tag 54
Symbol	The contract identifier (Tag 55) - e.g., BFUT500-ERCOT-HB_NORTH-260122-15
Time In Force	How long an order remains active (Tag 59): DAY, GTC, IOC, FOK
DAY	Day order (59=0) - remains active until end of trading session or canceled
GTC	Good 'Til Canceled (59=1) - remains active until filled or explicitly canceled
IOC	Immediate or Cancel (59=3) - fills what's available immediately, cancels remainder
FOK	Fill or Kill (59=4) - must fill completely immediately or be rejected entirely
CumQty	Cumulative Quantity (Tag 14) - total quantity filled so far

LeavesQty	Leaves Quantity (Tag 151) - remaining quantity still to be filled
LastPx	Last Price (Tag 31) - price of the most recent fill
LastQty	Last Quantity (Tag 32) - quantity of the most recent fill
MinQty	Minimum Quantity (Tag 110) - for IOC orders, minimum acceptable fill quantity
ResendRequest	A message (35=2) requesting replay of missed messages
PossDupFlag	A flag (Tag 43=Y) set when a message is being replayed, indicating a "possible duplicate"
SenderCompID	The sender's FIX identifier (Tag 49)
SenderSubID	User/trader identifier (Tag 50) - required for Order Entry to identify specific trader
TargetCompID	The recipient's FIX identifier (Tag 56) - EXI when sending to ElectronX
MsgSeqNum	Message Sequence Number (Tag 34) - the sequence number of this specific message
Cancel on Disconnect (CoD)	Automatic cancellation of DAY orders when you unexpectedly disconnect; GTC orders remain active
Cancel on Logout (CoL)	Automatic cancellation of DAY orders when you send Logout message; GTC orders remain active
BusinessMessageReject	A message (35=j) indicating business rule violation - valid FIX format but violates exchange rules.
Reject	A message (35=3) indicating protocol violation - invalid FIX message structure
Tick Size	Minimum price increment allowed (e.g., 0.25 for ElectronX Bounded Futures and Binary Options)

SecurityListRequest	Message (35=x) requesting list of available trading instruments
SecurityList	ElectronX's response message (35=y) listing available instruments and their specifications
SecurityGroup	Product family grouping (Tag 1151) - e.g., “BFUT100-ERCOT-HB_NORTH”

APPENDIX B: FURTHER READING

ElectronX Documentation:

- **FIX Specification** - Detailed message formats, required fields, enum values
- **QuickFix Setup Guide** - Configuration instructions for QuickFix libraries
- **FIX Reference Guide** - Basic information on FIX provisioning and access
- **Production API Trading Approval Framework** - Requirements for going live

Getting Help:

- **ElectronX Support:** support@electronx.com

Document Version: 1.0

Last Updated: January 2026
Feedback: support@electronx.com