**FIX** = Financial Information eXchange protocol — a standard messaging format used in the finance industry to communicate between trading systems (e.g., from your trading system to a broker or exchange).

## **FIX Message Structure: Overview**

A FIX message has **three main parts**:

1. **Header**
2. **Body**
3. **Trailer**

Each part contains a set of **tag=value** fields, separated by **SOH (ASCII 0x01)** — usually shown as | in examples for readability.

**Example: New Order for 100 AAPL shares @ $150.25**

8=FIX.4.2|9=112|35=D|49=BUY\_SIDE|56=SELL\_SIDE|34=3|52=20250322-14:25:00.000|

11=ABC123|21=1|55=AAPL|54=1|38=100|40=2|44=150.25|10=157|

## **HEADER SECTION**

This part is **required** in every FIX message and helps with **routing, versioning, and sequencing**.

| **Tag** | **Name** | **Description** |
| --- | --- | --- |
| 8 | BeginString | FIX version (e.g., FIX.4.2, FIX.4.4) |
| 9 | BodyLength | Length of everything after 9= up to 10= (excluding 10) |
| 35 | MsgType | Type of message (e.g., D = New Order, F = Cancel) |
| 34 | MsgSeqNum | Sequence number (for session tracking, increasing number) |
| 49 | SenderCompID | Sender’s ID (e.g., your system’s ID) |
| 56 | TargetCompID | Receiver’s ID (e.g., broker or exchange ID) |
| 52 | SendingTime | Time the message was sent, in UTC (YYYYMMDD-HH:MM:SS.sss) |

👉 **Purpose**: This section ensures the message is **correctly identified and routed**.

**BODY SECTION**

The **actual business data** (like the order details) is included here.

The fields here **vary based on MsgType (35)**.

### **Example: MsgType = D (New Order Single)**

| **Tag** | **Name** | **Description** |
| --- | --- | --- |
| 11 | ClOrdID | Client Order ID (unique for your orders) |
| 21 | HandlInst | Handling instruction (e.g., 1 = automated execution) |
| 55 | Symbol | Ticker symbol (e.g., AAPL) |
| 54 | Side | 1 = Buy, 2 = Sell |
| 38 | OrderQty | Quantity of shares/contracts |
| 40 | OrdType | 1 = Market, 2 = Limit |
| 44 | Price | Required if OrdType=2 (Limit) |
| 59 | TimeInForce | (optional) 0 = Day, 1 = GTC (Good Till Cancel), etc. |

👉 **Purpose**: This section carries the **action** you want the recipient to perform — like placing or canceling an order.

## **TRAILER SECTION**

This part is small but crucial for **message integrity**.

| **Tag** | **Name** | **Description** |
| --- | --- | --- |
| 10 | CheckSum | 3-digit checksum of all bytes from 8= to the last SOH before 10= |

**Purpose**: Validates that the message was **not corrupted in transmission**.

## **Message Lifecycle (Trading Flow)**

* Send **New Order** (35=D)
* Receive **Execution Report** (35=8) — Ack or Fill
* Send **Cancel** (35=F)
* Send **Replace** (35=G)

## **🔍 FIXatdl Schema Files (What Each Does)**

| **XSD File** | **Category** | **Purpose** |
| --- | --- | --- |
| **fixatdl-core-1-2.xsd** | Data | Defines base elements (parameters, strategies) — core of FIXatdl |
| **fixatdl-validation-1-2.xsd** | Data | Defines validation logic (min, max, comparisons, complex rules) |
| **fixatdl-layout-1-2.xsd** | GUI | Defines layout and UI controls (TextField, DropDownList, etc.) |
| **fixatdl-flow-1-2.xsd** | GUI | Defines dynamic behavior (e.g., show/hide based on user input) |
| **fixatdl-regions-1-2.xsd** | Data | Country and region enumerations (e.g., EuropeMiddleEastAfrica) |
| **fixatdl-timezones-1-2.xsd** | Data | List of valid timezones (used in parameters like StartTime) |

* [xsd2ts](https://www.npmjs.com/package/xsd2ts) or [quicktype.io](https://quicktype.io/) to convert .xsd schemas into TypeScript interfaces
* VS Code + XML plugins for schema-aware editing

Key Concepts

* The <Strategies> element contains multiple <Strategy> definitions, each representing a different algorithmic trading strategy, like VWAP, POV, etc.But when a user selects a strategy in a trading GUI, and the system sends a FIX message, how does the recipient know which strategy is being used? 👉 This is where the strategyIdentifierTag comes into play.
* In general, when discussing the parameters of an algorithmic order, one refers to the user-defined fields of a NewOrderSingle(35=D), OrderCancelRequest(35=F), and OrderCancelReplaceRequest(35=G) message.

Parameter Description

| **Item** | **Details** |
| --- | --- |
| **Used In Messages** | NewOrderSingle (35=D), Cancel (35=F), Replace (35=G) |
| **Defined By** | Broker (order recipient), shared to client via FIXatdl XML |
| **Element** | <Parameter> |
| **Attributes** | name, xsi:type, fixTag, use, minValue, maxValue |
| **Enum Support** | Use <EnumPair enumID="" wireValue=""/> for predefined choices |
| **Type** | xsi:type (e.g., Int\_t, Char\_t, UtcTimeOnly\_t) maps to FIX datatypes |
| **Tag Range** | Usually custom tags (>5000), e.g., 28000, 28001 |
| **Validation** | Done via min/max or enum constraints |

* **Algorithmic Order Interface**:
  + Defined by a set of FIX messages:
    - NewOrderSingle (35=D)
    - OrderCancelRequest (35=F)
    - OrderCancelReplaceRequest (35=G)
  + **Parameter = User-defined Field**
  + Can also include some standard FIX fields (1–5000), e.g.:
    - EffectiveTime (168)
    - ExpireTime (126)
  + **Defined By Broker (Order Recipient)**  Sent to clients (order initiators) via **FIXatdl XML**.
* **Parameter Element** Describes each input expected:
  + name: Unique within a strategy
  + xsi:type: FIX type (e.g. Int\_t, Char\_t)
  + fixTag: Tag number in FIX message (e.g. 28000)
  + use: required or optional
  + minValue / maxValue: For range validation
* **Enumerated Values (EnumPair)** Used when parameter allows predefined values.

### **Summary: Rendering Parameters in OMS Using FIXatdl®**

* **OMS Role**:  
   OMS (Order Management System) must render GUI controls to display strategy parameters.
* **Control Selection**:  
   GUI control type is selected based on parameter type:  
  + Price → Number spinner
  + Enum values (e.g., High/Medium/Low) → Combo box
* **Layout Handling**:  
   The **Layout Schema** in FIXatdl® defines how GUI controls are arranged on the screen.
* **Platform Neutral**:  
   FIXatdl® is **UI-style agnostic** and works across platforms (e.g., .NET, Java, Web).
* **StrategyPanel Element**:  
   Main container for arranging controls; supports nesting and alignment.  
    
   **Attributes:**
  + Title: Panel title (optional to display)
  + Collapsible: Whether the panel can be collapsed
  + Collapsed: Initial collapse state
  + Orientation: Alignment of controls (vertical, horizontal, or grid aligned)

### **Summary:**

| **Type of Control** | **Linked to Parameter?** | **Purpose** |
| --- | --- | --- |
| TextBox, Spinner, Dropdown | ✅ Yes | Input for FIX parameters |
| CheckBox, Button (UI logic) | ❌ No | Triggers behavior using <StateRule> |

Code Base

* **MainWindow.xaml** is the entry point and its DataContext is pointed to **MainViewModel**
* **MainViewModel** uses **ExampleStrategyProvider-** which is used for loading the strategies, providers, select strategy, etc.
* Passes the **Selected Strategy** to the **<AtdlControl>** WPF control.
* **AtdlControl.xaml.cs** has an implementation for **OnStrategyPropertyChanged** which effectively calls **Render()** which renders the strategy panel by dynamically building the *xaml* attaching **StrategyViewModel.**
* **StrategyViewModel** contains
* *public ViewModelControlCollection* ***Controls***
* *public ViewModelStrategyEditCollection* ***StrategyEdits***
* *private readonly Strategy\_t* ***\_underlyingStrategy;***

This is bound to the dynamically builds the ***XAML*** as shown below example  
  
<Label Grid.Column="0" Grid.Row="0" Target="{Binding ElementName=c\_Clock}"

IsEnabled="{***Binding Path=Controls[c\_Clock].Enabled***}"

Visibility="{Binding Path=Controls[c\_Clock].Visibility}" Content="Clock\_\_t:" />

* **ViewModelControlCollection** 
  + **-** contains a collection of ControlViewModel
  + - contains collection of StrategyEditViewModel