Tradias FIX Trading API

Introduction

Tradias offers FIX-based APIs for its customers. This document describes the FIX protocol-based Trading API and is written in the style of the FIX rules of engagement (RoE) document. Therefore, it represents a supplement to the official FIX specification published on the FPL website. Tradias may update this document anytime to reflect changes in our FIX Pricing API.

Tradias Trading Model

Tradias is a leading player in the OTC crypto market, setting new standards for liquidity and execution quality. Operating 24/7, Tradias offers exceptional crypto asset liquidity at competitive prices with minimal slippage, which is especially beneficial for large volume trades. As a principal desk, Tradias facilitates direct trades against its OTC desk, eliminating intermediaries and ensuring efficient transactions.

With a diverse range of over 150 crypto assets available for trading, Tradias supports various fiat currencies and stablecoins, offering a comprehensive trading experience tailored to institutional clients. Our robust APIs enables clients to automate trading, manage order flows, and perform seamless reconciliation and settlement. The platform supports over 200 trades per second per client, ensuring rapid execution and outstanding performance.

Pricing at Tradias is highly competitive and confidential, allowing clients to execute their strategies with discretion. To maintain transparency, it provides market conformity checks and detailed trading reports. Operating under the regulations of a certified credit institution in Germany, Tradias ensures minimal counterparty risk, making it a trustworthy partner for institutional trading.

Settlement processes are designed to be flexible and efficient, with options for T+1 and weekly cycles for post-trade settlements. Clients can settle via fiat methods, including SEPA and international wire transfers, or through custodial wallets for crypto transactions.

By providing a fully regulated infrastructure, Tradias opens the door for institutional investors to tap into the vast potential of digital assets, complemented by additional services such as tokenization and consulting.

All orders within the Tradias OTC platform are executed as Fill or Kill (FoK) orders, ensuring that orders are either fully executed or completely rejected, thereby simplifying order management. While limit orders can be placed, only FoK limit orders are supported currently.

FIX Protocol

Session level and connectivity rules

Network Options

The connectivity to the Tradias FIX Trading API is done over the Internet using TLS. Tradias is the server side and provides the following environments:

- Production
- Pre-Production

Tradias provides a list (usually two) of connection points (Hostname and Port) and a TLS certificate for each environment. To be accepted as a FIX client, the client's IP address must be whitelisted by Tradias. Please contact info-api@tradias.de for questions or to begin the onboarding process.

FIX Versions

The Tradias FIX Trading API is based on the FIX.4.4 protocol, available at FIX 4.4 Specification.

Encryption Support

Tradias uses a TLS-encrypted TCP connection over the Internet. To establish a connection, Tradias needs to whitelist the client's IP address and provide the client with an SSL certificate. There is no additional encryption in the FIX messages used!

Duplicate and Resend Message handling

The Tradias FIX Trading API uses the normal FIX sequence number mechanism to guarantee message delivery and avoid duplicates. In addition to this mechanism, critical requests like the New Order Single check the client-provided unique ClOrdID and detect duplicate messages based on it.

Start of Day / End of Day Procedures

The Tradias FIX Trading API acts as a server and expects clients to establish the connection. Tradias provides at least two FIX server addresses per the FIX connection. In case of a connection problem, the client should try any of the other addresses to reestablish the connection. The FIX sessions of the Trading API use message recovery. Therefore, a technical disconnect will be recovered after reconnection and Logon without a loss of messages.

The connection time of a FIX session is 00:00 UTC until 23:59:59 UTC every day (incl. weekends and holidays). That implies a termination of the FIX sessions at midnight UTC. After the next connect and Logon, a sequence number reset will be forced from the Tradias side.

Counter Party Identification

Tradias supports multiple clients, while the FIX Trading API supports only one client per FIX session. Therefore, messages from and to different end clients in a single FIX session are not allowed.

The FIX Trading API identifies the client and Tradias by the fields SenderCompID (49) and TargetCompID (56). Messages sent by the client to the FIX Trading API need to fill the SenderCompID (49) with the client_id of the client (a UUID provided by Tradias during the onboarding process), and the TargetCompID (56) will be filled with the tradias_id (a UUID provided by Tradias during the onboarding process). For messages sent by Tradias to the client, Tradias fills the SenderCompID (49) with the tradias_id (a UUID provided by Tradias during the onboarding process) and the TargetCompID (56) with the client_id of the client (a UUID provided by Tradias during the onboarding process).

FIX Header and Trailer

The FIX Trading API uses the following FIX Header:

Field	Name	Type	Req	Comment
8	BeginString	String	✓	Must be first field in message (“FIX.4.4”).
9	BodyLength	Length	✓	Must be second field in message.
35	MsgType	String	✓	Must be third field in message.
49	SenderCompID	String	✓	For Messages sent by the client: client_id, otherwise tradias_id. Both are provided by Tradias during the onboarding.
56	TargetCompID	String	✓	For Messages sent by the client: tradias_id, otherwise client_id. Both are provided by Tradias during the onboarding.
34	MsgSeqNum	SeqNum	✓	Integer message sequence number.
52	SendingTime	UTCTimestamp	✓	Time of message transmission, always expressed in UTC.

The FIX Trailer of the FIX Trading API is the following:

Field	Name	Type	Req	Comment
10	CheckSum	String	✓	Three-byte, simple checksum. See the FIX spec for details. ALWAYS LAST FIELD IN MESSAGE, i.e. serves, with the trailing <SOH>, as the End-Of-Message delimiter. Always defined as three characters.

FIX Session Level Messages

The FIX Trading API uses the following session-level messages. For further details, see the FIX spec.

Logon [A]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = A
98	EncryptMethod	int	✓	Method of encryption. Must be 0 = None.
108	HeartBtInt	String	✓	Heartbeat interval (seconds). Note same value used by both sides.
141	ResetSeqNumFlag	Boolean	—	Indicates that both sides of the FIX session should reset sequence numbers.
	Trailer	Component	✓	

Heartbeat [0]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = 0
112	TestReqID	String	—	Required when the heartbeat is the result of a Test Request message.
	Trailer	Component	✓	

Test Request [1]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = 1
112	TestReqID	String	✓	Identifier included in Test Request message to be returned in resulting Heartbeat.
	Trailer	Component	✓	

Resend Request [2]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = 2
7	BeginSeqNo	SeqNum	✓	Message sequence number of first message in range to be resent.
16	EndSeqNo	SeqNum	✓	Message sequence number of last message in range to be resent. If request is for a single message BeginSeqNo (7) = EndSeqNo (16). If request is for all messages subsequent to a particular message, EndSeqNo (16) = "0" (representing infinity).
	Trailer	Component	✓	

Reject [3]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = 3
45	RefSeqNum	SeqNum	✓	MsgSeqNum of rejected message.
371	RefTagID	int	—	The tag number of the FIX field being referenced.
372	RefMsgType	String	—	The MsgType of the FIX message being referenced.

373	SessionRejectReason	int	—	Code to identify reason for a session-level Reject message.
58	Text	String	—	Where possible, message to explain reason for rejection.
	Trailer	Component	✓	

Sequence Reset [4]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = 4
123	GapFillFlag	Boolean	—	Indicates that the Sequence Reset message is replacing administrative or application messages which will not be resent.
36	NewSeqNo	SeqNum	✓	New sequence number.
	Trailer	Component	✓	

Logout [5]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = 5
58	Text	String	—	Indicates that the Sequence Reset message is replacing administrative or application messages which will not be resent.
	Trailer	Component	✓	

Failover Procedures

To make the Tradias FIX service resilient, Tradias provides a list of server addresses (Hostname and Port). The client can use any of them to connect. If an address does not allow a connection, the client should try another provided address until a connection is possible. When the connection is reestablished, and the client has completed the Logon, sequence numbers must be synced (resend any messages generated but not received previously).

Definition of Security ID Conventions

In the Tradias FIX Trading API, the crypto pair/token is identified by the tag Symbol (55). The content is the Identifier of the instrument. It consists of the asset codes of the two legs and an optional asset tenor: {leg_1_asset} “-” {leg_2_asset} [“/” {tenor}]

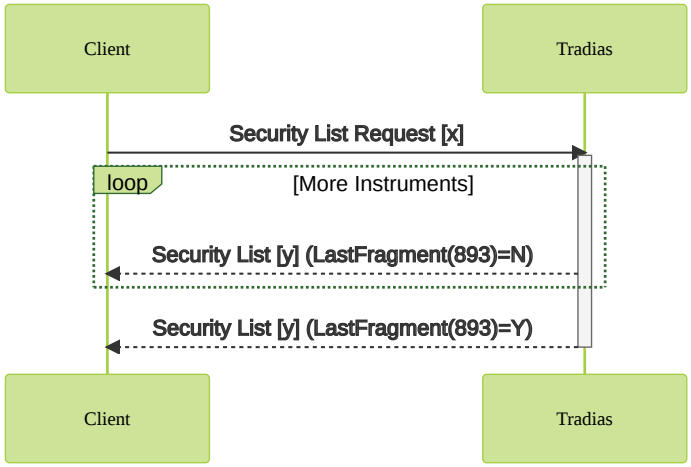
The tenor can be skipped if it is SP (spot).

Example (Bitcoin against Euro spot): BTC-EUR/SP or BTC-EUR

Business message types

Static Data

The client can request a list of tradeable instruments with the “Security List Request” [x]. Tradias will answer with a sequence of “Security List” [y] messages containing tradeable instruments. The last “Security List” [y] message will have set LastFragment (893) to “Y”.



Security List Request [x]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = x
320	SecurityReqID	String	✓	Unique ID of a Security Definition Request.
559	SecurityListRequestType	int	✓	Type of Security List Request being made. Valid values: 4 = All Securities
	Trailer	Component	✓	

Security List [y]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = y
320	SecurityReqID	String	✓	Unique ID of the related Security Definition Request.
322	SecurityResponseID	String	✓	Identifier for the Security List message.
560	SecurityRequestResult		✓	Result of the Security Request identified by the SecurityReqID. Valid values: 0 = valid request 1 = invalid or unsupported request 2 = no instruments found that match selection criteria 3 = not authorized to retrieve instrument data 4 = instrument data temporarily unavailable 5 = request not supported
393	TotNoRelatedS	int	—	Used to indicate if the total number of securities being returned for this request. Used in the event that message fragmentation is required.
893	LastFragment	Boolean	—	Indicates whether this message is the last in a sequence of messages. Valid values: Y = last message N = not last message
146	NoRelatedSym	NumInGroup	—	Specifies the number of repeating symbols specified in this message.
55	Symbol	String	✓	Identifier of the instrument. It consists of the asset codes of the two legs and an optional asset tenor: {leg_1_asset} “-” {leg_2_asset} [“/” {tenor}]

				Example (Bitcoin against Euro spot): BTC-EUR/SP
969	MinPriceIncrement	float	✓	Minimum price increment for the instrument. FIX 5.0 Field
561	RoundLot	Qty	✓	Minimum size increment for the instrument.
562	MinTradeVol	Qty	–	The minimum order quantity for the instrument.
1140	MaxTradeVol	Qty	–	The maximum order quantity for the instrument. FIX 5.0 SP1 Field
	Trailer	Component	✓	

Pricing

Tradias offers a separate FIX Pricing API to subscribe to the actual prices for any tradeable instrument. The subscription has to be for aggregated Bid and Offer prices. Tradias will provide prices for all agreed-upon sizes. Please have a look to the documentation of the FIX Pricing API.

Trading

An order is a request for buying or selling a specific asset. Order requests always specify the trading pair, quantity, and side at which the trader is willing to transact. For limit orders, the price should also be given.

It is possible to place both market orders and limit orders via our API. The implementation of limit orders is restricted in the sense that we only allow FOK/IOC (Fill or kill/Immediate or cancel) orders.

Types of Orders:

- **Market Order:** In a market order, the API user at least specifies the instrument, quantity and side (buy or sell). Execution of market orders is done immediately, and the execution price is the newest price for the corresponding instrument, quantity and side.
- **Limit Order:** In a limit order, the client specifies a price they are willing to buy or sell. Next, the client has to specify TimeInForce (59), which has to be either “FOK” or “IOC.” Our current implementation of limit orders is restricted to FOK/IOC orders. This means that limit orders

are not kept in an order book; they are either executed immediately or cancelled immediately, depending on the parameters in the order request.

- Immediate or Cancel (IOC) and Fill or Kill (FOK): when placing a limit order, the API client has to enter either “IOC” or “FOK” in the TimeInForce (59) field. The order will either be executed or rejected immediately. The order is accepted when:
  - Price (44) >= actual Tradias price, and the order is a buy order.
  - Price (44) <= actual Tradias price, and the order is a sell order.

In all other cases, the limit order will be rejected immediately. In case of an execution, **the limit order is executed at the actual Tradias price, not the limit price**. This means that executions are always done at a price that is the same or better than the specified Price (44). Note that there is no difference in how “FOK” or “IOC” limit orders are handled.

The answer to New Order Single [D] sent to Tradias is an Execution Report [8]. With that message, Tradias informs the client about the execution of the order. It will either report the rejection or non-execution of the order and explain the reason for such, or it will report the execution of the order with details about the execution (e.g., the price).

New Order Single [D]

Field	Name	Type	Req	Comment
	Header	Component	✓	MsgType (35) = D
11	ClOrdID	String	✓	Unique identifier of the order as assigned by the client. Uniqueness must be guaranteed within a client. Will be converted by Tradias into the order_ref an the req_id field. This ensures that an order can only be executed once, even if it is sent multiple times.
526	SecondaryClOrdID	String	–	Alternative identifier for the order. Can be freely assigned by the client for an order. Will be converted by Tradias into an user_data element with tag “SecondaryClOrdID”.
583	ClORdLinkID	String	–	Permits order originators to tie together groups of orders in which trades resulting from orders.

				Will be converted by Tradias into an user_data element with tag “ClOrdLinkID”.
58	Text	String	—	Free format text string. Will be converted by Tradias into an user_data element with tag “Text”.
55	Symbol	String	✓	Identifier of the instrument. It consists of the asset codes of the two legs and an optional asset tenor: {leg_1_asset} “-” {leg_2_asset} [“/” {tenor}] Example (Bitcoin against Euro spot): BTC-EUR/SP
54	Side	char	✓	Side of the order from the perspective of the client. Valid values: 1 = Buy 2 = Sell
40	OrderType	char	—	Order type. If not specified, the default is 1=Market Valid values: 1 = Market - Default 2 = Limit
38	OrderQty	Qty	—	Size of the order expressed in the leg_1_asset (BTC in case of BTC-EUR/SP). One of the two fields OrderQty (38) and CashOrderQty (152) is mandatory. If none or both are given, an error is reported.
152	CashOrderQty	Qty	—	Size of the order expressed in the leg_2_asset (EUR in case of BTC-EUR/SP). One of the two fields OrderQty (38) and CashOrderQty (152) is mandatory. If none or both are given, an error is reported.
44	Price	Price	—	Required for OrderType (40) 2=Limit.

59	TimeInForce	char	—	Specifies how long the order remains in effect. Absence of this field is interpreted as 3=Immediate or Cancel (IOC). Valid values: 3 = Immediate or Cancel (IOC) - Default 4 = Fill or Kill (FOK)
2215	LegInstrumentRoundingPrecision	int	—	Specifies the rounding precision in terms of a number of decimal places for the leg where the quantity has not been given in the order. If OrderQty (38) is given this parameter specifies the precision of the leg_2_quantity and if CashOrderQty (152) is given, the precision of the leg_1_quantity is defined. If not given the precision is assumed to be 0 which is interpreted as maximal precision of the leg_asset. FIX 5.0 SP2 Field
60	TransactTime	UTCTimestamp	✓	Time this order request was initiated/released by the trader, trading system, or intermediary (expressed in UTC (Universal Time Coordinated, also known as "GMT").
	Trailer	Component	✓	

Execution Report [8]

This message is used to:

- Relay fill information on received orders
- Reject orders

Each execution report contains two fields, which communicate both the current state of the order as understood by Tradias (OrdStatus (39)) and the purpose of the message (ExecType (150)).

Field	Name	Type	Req	Comment

	Header	Component	✓	MsgType (35) = 8
37	OrderID	String	✓	Unique identifier of the order assigned by Tradias.
11	ClOrdID	String	✓	Unique identifier of the order as got assigned by the client.
526	SecondaryClOrdID	String	–	Alternative identifier for the order. Can be freely assigned by the client for an order. Will be converted by Tradias into an user_data element with tag “SecondaryClOrdID”.
583	ClORdLinkID	String	–	Permits order originators to tie together groups of orders in which trades resulting from orders. Will be converted by Tradias into an user_data element with tag “ClOrdLinkID”.
58	Text	String	–	Free format text string. Will be converted by Tradias into an user_data element with tag “Text”.
55	Symbol	String	✓	Identifier of the instrument. It consists of the asset codes of the two legs and an optional asset tenor: {leg_1_asset} “-” {leg_2_asset} [“/” {tenor}] Example (Bitcoin against Euro spot): BTC-EUR/SP
54	Side	char	✓	Side of the order from the perspective of the client. Valid values: 1 = Buy 2 = Sell
40	OrderType	char	–	Order type. If not specified, the default is 1=Market Valid values: 1 = Market - Default

				2 = Limit
38	OrderQty	Qty	—	Size of the order expressed in the leg_1_asset (BTC in case of BTC-EUR/SP). One of the two fields OrderQty (38) and CashOrderQty (152) is mandatory. If none or both are given, an error is reported.
152	CashOrderQty	Qty	—	Size of the order expressed in the leg_2_asset (EUR in case of BTC-EUR/SP). One of the two fields OrderQty (38) and CashOrderQty (152) is mandatory. If none or both are given, an error is reported.
44	Price	Price	—	Required for OrderType (40) 2=Limit.
59	TimeInForce	char	—	Specifies how long the order remains in effect. Absence of this field is interpreted as 3=Immediate or Cancel (IOC). Valid values: 3 = Immediate or Cancel (IOC) - Default 4 = Fill or Kill (FOK)
2215	LegInstrumentRoundingPrecision	int	—	The rounding precision in terms of a number of decimal places for the leg where the quantity has not been given in the order. If OrderQty (38) is given this parameter specifies the precision of the leg_2_quantity and if CashOrderQty (152) is given, it specifies the precision of the leg_1_quantity. A value of 0 is interpreted as maximal precision of the leg_asset. FIX 5.0 SP2 Field
60	TransactTime	UTCTimestamp	✓	Time of the execution/rejection (expressed in UTC (Universal Time Coordinated, also known as "GMT").

17	ExecID	String	✓	Unique identifier of execution message as assigned by Tradias.
150	ExecType	char	✓	Describes the specific Execution Report while OrdStatus(39) will always identify the current order status. Valid values: 4 = Canceled 8 = Rejected F = Trade
39	OrdStatus	char	✓	Describes the current state of the order. Valid values: 2 = Filled 4 = Canceled 8 = Rejected
151	LeavesQty	Qty	✓	Quantity open for further execution. Always 0.
14	CumQty	Qty	✓	Total quantity filled
6	AvgPx	Price	✓	Calculated average price of all fills on this order.
103	OrdRejReason	int	—	For optional use with ExecType = 8 (Rejected). Valid values: 0 = Tradias option 1 = Unknown symbol 2 = Trading closed 3 = Order exceeds limit 5 = Unknown order 6 = Duplicate Order (e.g. dupe ClOrdID)
31	LastPx	Price	—	Price of this fill. Required if ExecType(150) = F
32	LastQty	Qty	—	Quantity bought/sold on this fill. Required if ExecType(150) = F
381	GrossTradeAmt	Amt	—	Last amount traded (i.e. quantity * price) expressed in units of the leg_2. Required if ExecType(150) = F

	Trailer	Component	✓	

Test scripts

To define test cases for the Tradias FIX-based API described in the text, we need to create scenarios that validate the correct behaviour of the system under various conditions. Here are some sample test cases categorized by functionality:

Connectivity and Session Management

1. **Connectivity Test**
   - **Objective**: To ensure the client can connect to the Tradias server over the internet using TLS.
   - **Steps**:
     - Attempt to establish a connection using the provided hostname and port.
     - Validate that the connection is secured using the provided TLS certificate.
   - **Expected Result**: The connection is successfully established using TLS.
2. **Whitelist IP Test**
   - **Objective**: To verify that only whitelisted IP addresses are accepted.
   - **Steps**:
     - Attempt to connect to the server using a non-whitelisted IP address.
   - **Expected Result**: Connection should be refused.
3. **Login Session Test**
   - **Objective**: To confirm successful logon and logoff procedures.
   - **Steps**:
     - Send a Logon (Message Type A) with the correct credentials.
     - Send a Logout (Message Type 5) to terminate the session.
   - **Expected Result**: Login and logout should succeed without errors.

Static Data Handling

1. **Instrument List Request Test**
   - **Objective**: To confirm the client can request and receive a list of tradable instruments.
   - **Steps**:
     - Send a Security List Request [x] to request tradable instruments.
     - Receive a series of "Security List" [y] messages containing tradable instruments.

- **Expected Result**: The final "Security List" [y] message should have LastFragment (893) set to "Y", indicating the end of the list.

Trading

1. **Market Order Placement Test**

- **Objective:** To ensure a market order is placed successfully.
  Steps:
  - Send a New Order Single [D] message specifying:
    - ClOrdID (unique identifier)
    - Symbol (BTC-EUR)
    - Side (1 for Buy)
    - OrderQty (quantity, e.g., 1 BTC)
  - **Monitor** the Execution Report [8] response.
  - **Expected Result:** The Execution Report should indicate a successful execution with the status `OrdStatus` (39) as `2` (Filled). The `CumQty` should match the `OrderQty` specified.

2. **Limit Order Placement (Fill or Kill) Test**

- **Objective:** To confirm that a limit order (FOK) is either filled or cancelled immediately.
  Steps:
  - Send a New Order Single [D] message specifying:
    - ClOrdID
    - Symbol (BTC-EUR)
    - Side (2 for Sell)
    - OrderQty
    - Price (set above current market price)
    - TimeInForce (4 for Fill or Kill)
  - **Monitor** the Execution Report [8] response.
  - **Expected Result:** Since the price is above the current market price, the order should be rejected with `OrdStatus` (39) as `8` (Rejected) and `OrdRejReason` indicating the reason (e.g., unknown symbol or price issues).

3. **Limit Order Placement (Immediate or Cancel) Test**

- **Objective:** To ensure that an IOC limit order is executed appropriately.
  Steps:

- Send a New Order Single [D] message with:
  - ClOrdID
  - Symbol (BTC-EUR)
  - Side (1 for Buy)
  - OrderQty
  - Price (set at or below current market price)
  - TimeInForce (3 for Immediate or Cancel)
- **Monitor** the Execution Report [8].
- **Expected Result:** The order should be executed immediately, with `OrdStatus` (39) as `2` (Filled) and corresponding execution details provided.

4. **Duplicate Order Handling Test**

- **Objective:** To verify that duplicate orders are correctly identified and rejected.
  Steps:
  - Send a New Order Single [D] message with the same ClOrdID multiple times.
  - **Monitor** the Execution Report [8] responses.
  - **Expected Result:** The first order is executed successfully, but subsequent orders with the same ClOrdID should be rejected with `OrdStatus` (39) as `8` (Rejected) and `OrdRejReason` indicating "Duplicate Order".

5. **Order with Missing Required Fields Test**

- **Objective:** To confirm that orders lacking mandatory fields are rejected.
  Steps:
  - Send a New Order Single [D] message without the required OrderQty or CashOrderQty fields.
  - **Monitor** the Execution Report [8].
  - **Expected Result:** The order should be rejected with `OrdStatus` (39) as `8` (Rejected) and a suitable rejection reason in `OrdRejReason`.

6. **Invalid Forex Pair Test**

- **Objective:** To ensure the system rejects orders for non-tradable instruments.
  Steps:
  - Send a New Order Single [D] message with a Symbol that does not exist (e.g., ABC-XYZ).
  - **Monitor** the Execution Report [8].

- **Expected Result:** The message response should indicate rejection with `OrdStatus` (39) as `8` (Rejected) and `OrdRejReason` as "Unknown symbol".

7. **Order Modification Scenario Test**

- **Objective:** To verify that the appropriate error is returned when attempting to modify an order that can't be modified.
  Steps:
  - Place a limit order with ClOrdID.
  - Attempt to modify the same order once it is filled.
  - **Monitor** the execution report.
  - **Expected Result:** The response should indicate the modification attempt is rejected, with an appropriate `OrdRejReason` for the rejection.

8. **High Volume Order Handling Test**

- **Objective:** To evaluate how well the system manages high volume orders.
  Steps:
  - Send multiple New Order Single [D] messages in quick succession exceeding the 200 trades/second limit.
  - **Monitor** the Execution Reports [8].
  - **Expected Result:** The system should properly queue or reject orders based on the rate limit policy, with appropriate rejection messages for those that exceed the limit.

9. **Settlement Timing Test (T+1 and Weekly)**

- **Objective:** To ensure the client can select and execute orders with proper settlement timing.
  Steps:
  - Send multiple New Order Single [D] messages specifying settlement terms for T+1 and weekly.
  - **Monitor** the Execution Reports [8] for successful execution.
  - **Expected Result:** The system should accept valid settlement terms, and appropriate confirmations should be reflected in the execution report.

10. **Error Logging on Failed Order Placement**

- **Objective:** To check that error responses for failed order placements are logged accordingly.
  Steps:
  - Attempt to place an order with an invalid price format.
  - **Monitor** the logs and Execution Reports [8].

- **Expected Result:** The system should log the error with relevant details and notify the user via the Execution Report of the rejection.

Failover and Recovery

1. **Server Failover Test**
   - **Objective**: To verify client failover procedures to secondary server addresses in case of primary server failure.
   - **Steps**:
     - Simulate a primary server failure.
     - Attempt to connect to a secondary server address.
   - **Expected Result**: The client should successfully connect to the secondary server and continue operations.

These test cases cover a range of scenarios that the Tradias FIX-based API may encounter during operation. They are designed to ensure the system is robust, reliable, and follows the FIX protocol standards.

Examples

Logon

Logon Request

In the context of FIX 4.4, a logon message is denoted by the MsgType (35) set to "A". Here's an example of a logon message with randomly generated UUIDs for the SenderCompID and TargetCompID:

```
8=FIX.4.4|9=133|35=A|34=1|49=123e4567-e89b-12d3-a456-426614174000|56=123e4567-e89b-12d3-a456-426614174001|52=20240813-16:20:00.000|98=0|108=30|141=Y|10=124|
```

In the above message:

- `8=FIX.4.4` specifies the FIX version.
- `9=133` is the BodyLength, which must be calculated as the character count of the entire message excluding the start (8) and end (10) tags.
- `35=A` indicates it's a logon message.
- `34=1` is the MsgSeqNum, indicating this is the first message in the sequence.

- `49=123e4567-e89b-12d3-a456-426614174000` is the SenderCompID, a randomly generated UUID for the sender.
- `56=123e4567-e89b-12d3-a456-426614174001` is the TargetCompID, a randomly generated UUID for the receiver.
- `52=20240813-16:20:00.000` is the SendingTime in UTC, formatted as YYYYMMDD-HH:MM:SS.sss.
- `98=0` represents the EncryptMethod, set to 0, indicating no encryption.
- `108=30` is the HeartBtInt, representing the heartbeat interval in seconds.
- `141=Y` indicates that the sequence number should be reset.
- `10=124` is the CheckSum, which should sum all bytes in the message modulo 256.

Please note that the BodyLength (9) and the CheckSum (10) need to be calculated correctly. The vertical bar `|` is used here for clarity but should be replaced with the ASCII character for "Start of Heading" (SOH), `0x01`.

Also, the UUIDs are just examples; in a real-world scenario, Tradias provides those during onboarding.

Logon Response

The logon response is typically a mirrored version of the logon request with some additional fields that may be specific to the session. Below is an example of how a logon response to the above message might look like, with a placeholder for the calculated checksum, assuming the server has accepted the logon request:

> 8=FIX.4.4|9=133|35=A|34=2|49=123e4567-e89b-12d3-a456-426614174001|56=123e4567-e89b-12d3-a456-426614174000|52=20240813-16:21:00.000|98=0|108=30|141=N|10=115|

In this response:

- `8=FIX.4.4` specifies the FIX version.
- `9=133` is the BodyLength, which must be calculated for the response.
- `35=A` indicates it's a logon message.
- `34=2` is the MsgSeqNum, indicating this is the second message in the sequence (the response).
- `49=123e4567-e89b-12d3-a456-426614174001` is the SenderCompID, which now contains the server's UUID, as it is the sender of the response.

- `56=123e4567-e89b-12d3-a456-426614174000` is the TargetCompID, which now contains the client's UUID, as it's the intended recipient.
- `52=20240813-16:21:00.000` is the SendingTime in UTC, formatted as YYYYMMDD-HH:MM:SS.sss, indicating the time the server sends the response.
- `98=0` represents the EncryptMethod, set to 0, indicating no encryption.
- `108=30` is the HeartBtInt, representing the heartbeat interval in seconds agreed upon.
- `141=N` indicates that the sequence number should not be reset, as this is a continuous session.
- `10=115` is the CheckSum, which should be calculated based on the content of the message.

Like with the logon request, the BodyLength (9) and the CheckSum (10) fields must be calculated accurately to validate the message. The vertical bar `|` is used here for readability but should be replaced with the SOH character in a real FIX message. The sequence number and the timestamp should be adjusted to match the actual values at the response time.

Static data request (Instrument List)

Request

Here is a sample FIX message requesting an instrument list and the corresponding response based on FIX 4.4 protocol specifications.

```
8=FIX.4.4|9=136|35=x|34=3|49=123e4567-e89b-12d3-a456-426614174000|56=123e4567-e89b-12d3-a456-426614174001|52=20240813-16:40:00.000|320=REQID67890|559=4|10=223|
```

In this request:

- `35=x` indicates it's a Security List Request.
- `320=REQID67890` is the SecurityReqID, a unique identifier for the security list request.
- `559=4` is the SecurityListRequestType, with the value `4` indicating a request for all securities.
- `9=136` and `10=223` are the BodyLength and CheckSum, which need to be calculated.

Response

```
8=FIX.4.4|9=294|35=y|34=4|49=123e4567-e89b-12d3-a456-426614174001|56=123e4567-e89b-12d3-a456-426614174000|52=20240813-
```

16:40:05.000|320=REQID67890|322=RESPID12345|560=0|393=2|893=Y|146=2|55=BTC-EUR|969=0.000000001|561=0.000001|562=0.000001|1140=20|55=ETH-EUR|969=0.000000001|561=0.000001|562=0.000001|1140=2000|10=135|

In this response:

- `35=y` indicates it's a Security List message.
- `320=REQID67890` corresponds to the original SecurityReqID of the request.
- `322=RESPID12345` is the SecurityResponseID, a unique identifier for the security list message.
- `560=0` is the SecurityRequestResult, indicating a valid request.
- `393=2` is the TotNoRelatedSym, indicating the total number of securities returned.
- `893=Y` is the LastFragment, indicating this is the last message in a sequence.
- `146=2` is the NoRelatedSym, specifying the number of symbols in the message.
- `55=BTC-EUR` and `55=ETH-EUR` are symbols for each security.
- `969=0.0001` and `969=0.01` are MinPriceIncrement for each symbol.
- `562=100000` is the MinTradeVol for each symbol.
- `9=294` and `10=135` are the BodyLength and CheckSum, which must be calculated.

Please note that in these messages:

- `|` represents the ASCII character for "Start of Header" (SOH), which is `0x01`.
- The BodyLength (`9`) value should be the character count of the message excluding the `8`, `9`, and `10` tags.
- The CheckSum (`10`) value is the sum of all ASCII values of the characters in the message modulo 256, which should be a three-digit number.

New Order and Execution Report

This examples is a FIX messages for a new order and the corresponding execution report. These messages are based on the FIX 4.4 protocol.

New Order Single Message

8=FIX.4.4|9=197|35=D|34=5|49=123e4567-e89b-12d3-a456-426614174000|56=123e4567-e89b-12d3-a456-426614174001|52=20240813-

16:40:10.000|11=O-08154711|55=BTC-EUR|54=1|40=2|38=0.34|44=20000|59=4|60=20240813-16:40:09.500|10=253|

In this order:

- `8=FIX.4.4` specifies the FIX version.
- `9=150` is the BodyLength, which must be calculated for the response.
- `35=D` indicates it's a New Order Single message.
- `34=5` is the MsgSeqNum, indicating this is the fifth message in the sequence.
- `49=123e4567-e89b-12d3-a456-426614174000` is the SenderCompID, a randomly generated UUID for the sender.
- `56=123e4567-e89b-12d3-a456-426614174001` is the TargetCompID, a randomly generated UUID for the receiver.
- `52=20240813-16:40:10.000` is the SendingTime in UTC, formatted as YYYYMMDD-HH:MM:SS.sss.
- `11=O-08154711` is the CIOrdID (unique identifier for the order).
- `55=BTC-EUR` is the Symbol of the trading pair.
- `54=1` is the Side (Buy).
- `40=2 is the OrderType (Limit).`
- `38=0.34` is the OrderQty (buying 0.34 BTC).
- `44=20000` is the Price (limit price per BTC in EUR).
- `59=4` ist the TimeInForce (Fill or Kill).
- `60=20240813-16:40:09.500` is the TransactTime in UTC, formatted as YYYYMMDD-HH:MM:SS.sss when the Trader has been initiated/released.
- `10=000` is the CheckSum, which should sum all bytes in the message modulo 256.

Execution Report Message (FIX Message Type 8)

8=FIX.4.4|9=289|35=8|34=6|49=123e4567-e89b-12d3-a456-426614174001|56=123e4567-e89b-12d3-a456-426614174000|52=20240813-16:40:39.000|37=94320047110815|11=O-08154711|55=BTC-EUR|54=1|40=2|38=0.34|44=20000|60=20240813-16:40:38.950|17=3494320047110815|150=F|39=2|151=0|14=0.34|6=20000|31=20000|32=0.34|381=6800|10=230|

In this execution report:

- `8=FIX.4.4` specifies the FIX version.
- `9=150` is the BodyLength, which must be calculated for the response.
- `35=8` indicates it's a Execution Report message.
- `34=6` is the MsgSeqNum, indicating this is the sixth message in the sequence (the response).
- `49=123e4567-e89b-12d3-a456-426614174001` is the SenderCompID, which now contains the server's UUID, as it is the sender of the response.
- `56=123e4567-e89b-12d3-a456-426614174000` is the TargetCompID, which now contains the client's UUID, as it's the intended recipient.
- `52=20240813-16:40:10.000` is the SendingTime in UTC, formatted as YYYYMMDD-HH:MM:SS.sss, indicating the time the server sends the response.
- `37=94320047110815` is the OrderID (unique identifier of the order assigned by Tradias)
- `11=0-08154711` is the CIOrdID (unique identifier for the order).
- `55=BTC-EUR` is the Symbol of the executed pair.
- `54=1` is the Side (Buy).
- `40=2` is the OrderType (Limit).
- `38=0.34` is the OrderQty (buying 0.34 BTC).
- `44=20000` is the Price (Limit).
- `60=20240813-16:40:38.950` is the TransactTime in UTC, formatted as YYYYMMDD-HH:MM:SS.sss, indicating the time the order has been executed/rejected.
- `17=3494320047110815` is the ExecID (unique identifier of execution message as assigned by Tradias).
- `150=F` ist the ExecType (Trade).
- `39=2` is the OrdStatus (Filled).
- `151=0` is the LeavesQty (quantity open for further execution).
- `14=0.34` is the CumQty (total quantity filled).
- `6=20000` is the AvgPx (average execution price).

- `31=20000` ist the LastPx (price of this fill).
- `32=0.34` is the LastQty (quantity of this fill).
- `381=6800` is the GrossTradeAmt (last amount traded i.e. quantity * price).
- `10=000` is the CheckSum, which should sum all bytes in the message modulo 256.

Please note that in these messages:

- `|` represents the ASCII character for "Start of Header" (SOH), which is `0x01`.
- The BodyLength (`9`) value should be the character count of the message excluding the `8`, `9`, and `10` tags.
- The CheckSum (`10`) value is the sum of all ASCII values of the characters in the message modulo 256, which should be a three-digit number.

Glossary

- **API (Application Programming Interface)**: A set of protocols and tools for building software applications, which, in this context, allows for interaction with the Tradias trading system.
- **BeginString (Tag 8)**: A FIX message field that indicates the beginning of a message.
- **BodyLength (Tag 9)**: A FIX message field that represents the length of a message body.
- **CheckSum (Tag 10)**: A FIX message field that provides a simple checksum to validate the integrity of message data.
- **FIX (Financial Information Exchange)**: A communication protocol for the real-time exchange of information related to securities transactions and markets.
- **FIX.4.4 Protocol**: A version of the FIX protocol that defines a specific set of message formats and rules for engaging in electronic financial transactions.
- **FPL (FIX Protocol Limited)**: The organization responsible for maintaining and developing the FIX protocol.
- **Heartbeat (Message Type 0)**: A FIX session-level message that confirms the connection is alive.
- **Logon (Message Type A)**: A FIX session-level message used when a user is logging on to the system.
- **Logout (Message Type 5)**: A FIX session-level message used to terminate a session.
- **Market Data Request (Message Type V)**: A FIX message type used to request market data.
- **Market Data Request Reject (Message Type Y)**: A FIX message used to reject a market data request.

- **Market Data Snapshot/Full Refresh (Message Type W)**: A FIX message used to provide a full snapshot of market data.
- **MsgSeqNum (Tag 34)**: A FIX message field that specifies the message sequence number.
- **MsgType (Tag 35)**: A FIX message field that defines the message type.
- **OTC (Over The Counter)**: Trading directly between two parties without a centralized exchange. This method is suitable for larger trades that could impact market prices if executed publicly.
- **Pre-Production Environment**: A testing environment that simulates the production environment.
- **Production Environment**: The live environment where the actual trading happens.
- **Reject (Message Type 3)**: A FIX session-level message used to reject a message that does not comply with the protocol.
- **RoE (Rules of Engagement)**: Guidelines or rules outlining how to interact with a particular FIX API.
- **SendingTime (Tag 52)**: A FIX message field that indicates the time a message was sent.
- **Session**: The period during which a user is logged on to a system.
- **Sequence Number**: A unique identifier for messages in a sequence that helps prevent message duplication.
- **SenderCompID (Tag 49)**: A FIX message field identifying the message's sender.
- **SSL Certificate**: A digital certificate that provides authentication for a website and enables an encrypted connection.
- **Symbol (Tag 55)**: A FIX message field representing the instrument's identifier.
- **TargetCompID (Tag 56)**: A FIX message field that identifies the message's intended recipient.
- **Test Request (Message Type 1)**: A FIX session-level message used to test the connection status.
- **TLS (Transport Layer Security)**: A cryptographic protocol designed to provide communications security over a computer network.
- **UUID (Universally Unique Identifier)**: A 128-bit number used to identify information in computer systems.
- **Whitelisting**: The process of allowing certain IP addresses to access a network or service, ensuring security.