Kalshi

FIX API Specifications (Version 1.0.10)

Contents

Change Log

2023-08-17

- Added drop copy session endpoint and message types

2023-10-19

- Added post trade session endpoint and message types
- Added position and cost related fields to Execution Report message type

2023-11-15

- Changes in Order Cancel Replace/ Order Modification (35=G) spec
    - Price change and OrderQty upwards change is now supported
    - More details added on how OrderQty in request is used
- Added the following fields to the Execution Report message type: Price, Symbol, OrigClOrdID

2023-11-17

- Added/updated descriptions for the following fields to the Execution Report message type: LastPx, LongQty, ShortQty, OrderQty

2024-01-26

- Added OrderMassCancelRequest spec

2024-04-09

- Added Security Group field to NewOrderSingle and OrderMassCancelRequest

2024-04-30

- Added SelfTradePreventionType field to NewOrderSingle

2024-08-22

- Updated CollateralType and SelfTradePreventionType field to ExecutionReport

2025-01-21

- Added RFQ session type and messages

2025-01-26

- Added market/event settlement message types

2025-04-15

- Removed deprecated event settlement message type
- Added ListenerSession and SkipPendingExecReports flag to Logon message type

Overview

FIX (Financial Information eXchange) is a standard protocol that can be used to enter orders, submit cancel requests, and receive fills.

Kalshi's implementation will follow the standards as much as possible, points where we had to diverge from the protocol will be highlighted in this document.

Connectivity

Before logging onto a FIX session, clients must establish a secure connection to the FIX gateway:

Purpose	Prod URL	Demo URL	Port	TargetCompID
Order Entry (without retransmission)	fix.elections.kalshi.com	fix.demo.kalshi.co	8228	KalshiNR
Order Entry (with retransmission)	fix-rt.elections.kalshi.com	fix-rt.demo.kalshi.co	8230	KalshiRT
Drop Copy	fix-dc.elections.kalshi.com	fix-dc.demo.kalshi.co	8229	KalshiDC
Post Trade	fix-rt.elections.kalshi.com	fix-pt.demo.kalshi.co	8231	KalshiPT
Rfq (with retransmission)	fix-rt.elections.kalshi.com	fix-rt.demo.kalshi.co	8232	KalshiRFQ

We will be providing the equivalent production stage endpoints as well as a market data endpoint in the near future.

Sessions are forcibly logged out every day between 2 AM ET and 2:10 AM ET for a maintenance window. All users are required to restart their sessions during this time and reset sequence numbers to 0.

Rate limits

There is a rate limit of 150 messages per second for the order entry session. This limit only counts application messages sent from the initiator (client) to the acceptor (server), **session layer messages will not count on this limit**.

Session layer messages ignored by the rate limit logic:

- Logon(35=A)
- Logout(35=5)
- HeartBeat(35=0)
- TestRequest(35=1)

Important observation: Specific limits per message type might be introduced in the future, but they will of course be communicated beforehand for proper adaptation on the initiator side.

TCP SSL

If your FIX implementation does not support establishing a native TCP SSL connection, you must set up a local proxy such as stunnel to establish a secure connection to our FIX gateway.

Important observation: Kalshi will provide the certificate for pinning on the initiator side when providing your api key.

Message Retransmission

Kalshi currently only supports retransmission on the order entry session with retransmission mentioned in the Connectivity section. This means that the following message types will not be supported in other endpoints:

- ResendRequest(35=2)
- SequenceReset (35=4)

For the session endpoints that do not support retransmission, ResetSeqNumFlag<141> in the Logon message should always be true or the Logon will be rejected.

The drop copy session endpoint provides an alternative way for clients to query for missed execution reports.

Message Headers

The baseline specification for this API is FIX 5.0 SP2, so we expected the session profile to always be **FIXT.1.1**. This is a requirement because we are making use of the Application Version Independence provided by FIXT.1.1 in order to progressively support multiple application layer versions.

A standard header must be present at the start of every message in both directions. You should configure your sessions so that

- SenderCompID = Fix Api Key (uuid). Create and copy a FIX API key from the demo website.
- TargetCompID = Please refer to the Connectivity section for the corresponding TargetCompID to use based on the endpoint you are trying to connect to

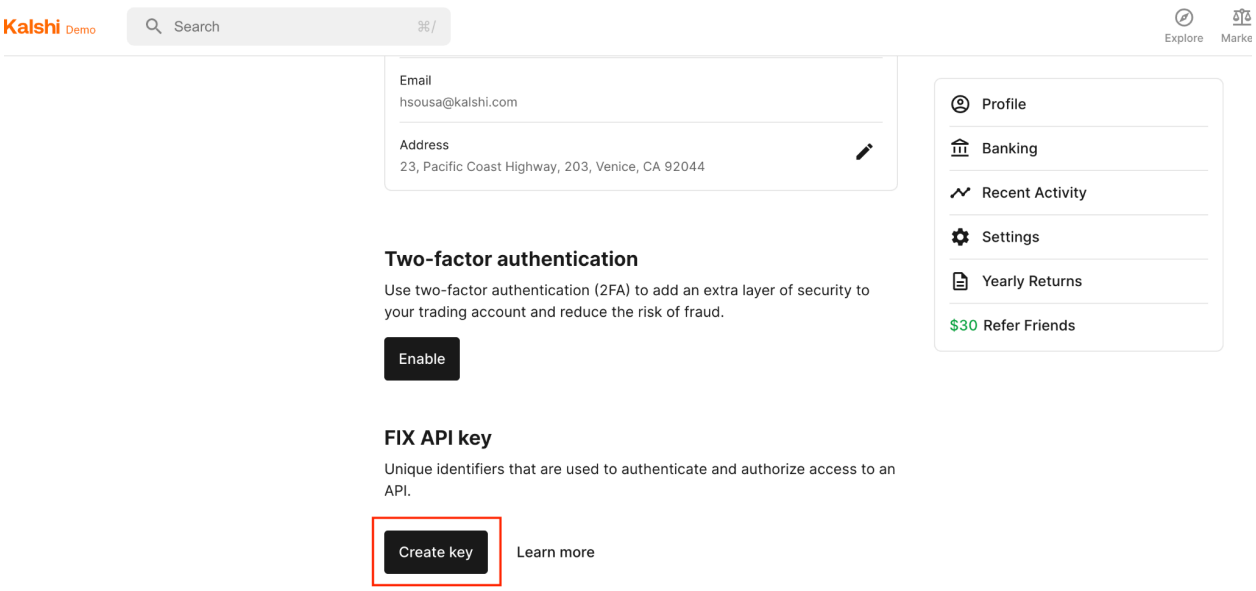
To create your FIX Api Key in the demo website you need to have a 2048 bit RSA PKCS#8 keys pair locally. You can generate it using the command below:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out kalshi-fix.key
openssl rsa -in kalshi-fix.key -pubout -out kalshi-fix.pub
```

The command above will generate two files in your file system:

- The private key: **kalshi-fix.key**. You should store this key securely; it is equivalent to your username + password. Should not be sent to anyone else, even to Kalshi employees. This key is what guarantees that you and only you can establish FIX connections with Kalshi on your behalf.
- The public key: **kalshi-fix.pub**. This key should be exchanged with Kalshi in order to create an Api Key.

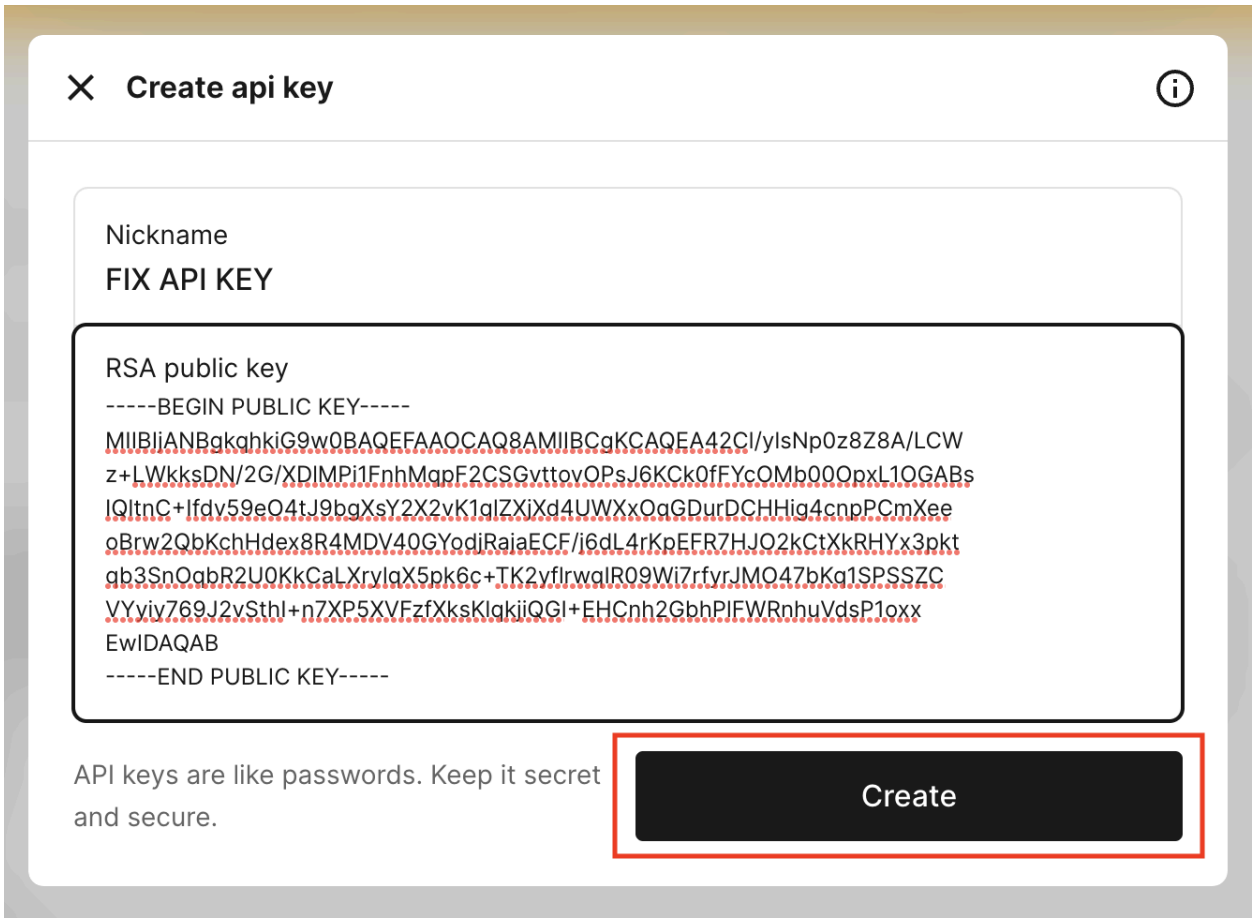
Then navigate to https://demo.kalshi.co/account/profile and click the 'Create key' button displayed below:



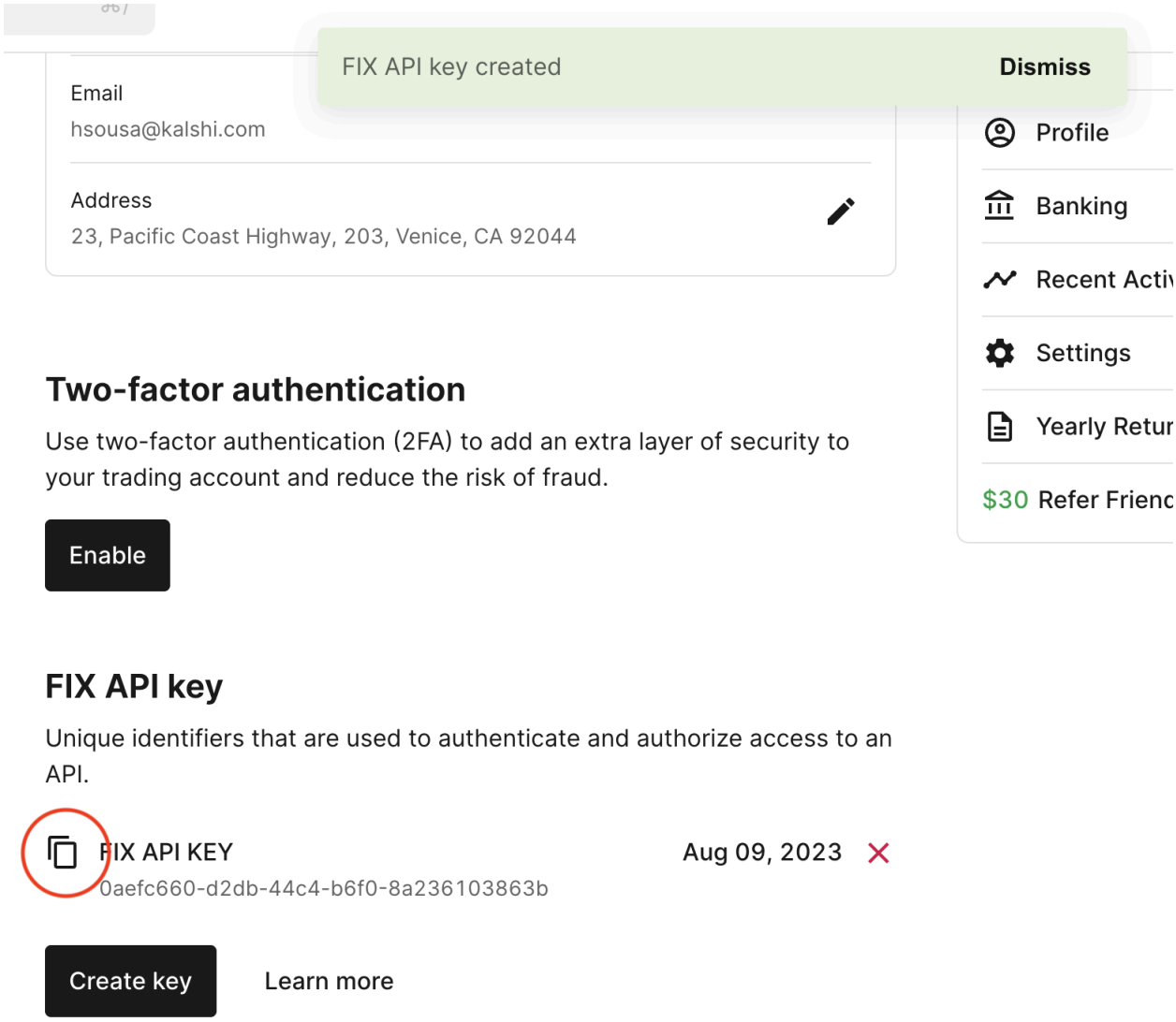
A dialog titled **"Create api key"** will appear, where you should:

- name your Api Key.
- copy and paste the contents of the kalshi-fix.pub key into the "**RSA public key"** field.

- click the **“Create”** button.



You should see a “FIX API key created” success snackbar after pressing the “Create” button. Like the one below:



You should click the copy icon circled in the screenshot above to copy your FIX Api Key value, in the example above that value would be: **0aefc660-d2db-44c4-b6f0-8a236103863b.**

Session identification

Session identification will be done using: **SessionID = TargetCompID + SenderCompID.**

Important observation: We are going to accept one FIX connection per FIX Api Key.

The configuration of SenderCompID and TargetCompID is typically accomplished via your FIX client's configuration file.

Tag	Name	Description	Data Type / Expected Value

Number			
8	BeginString ***Required**	Identifies the beginning of a message and version of the FIX session layer protocol. ***MUST BE THE FIRST FIELD IN THE MESSAGE***	String(enum) **FIXT.1.1**
9	BodyLength ***Required**	Length of the message that follows this field, in bytes. ***MUST BE THE SECOND FIELD IN THE MESSAGE***	Int Correct byte count as specified by FIX.
34	MsgSeqNum ***Required**	Message sequence number must be unique and increase by one unit for every message. ***WILL RESET EVERY DAY***	Int Correct NextNumOut for the initiator.
35	MsgType ***Required**	Defines the type of the message. ***MUST BE THE THIRD FIELD IN THE MESSAGE***	String(enum) One of the message types listed in this document.
52	SendingTime ***Required**	Time of the message transmission in UTC.	UTCTimestamp Timestamp within 120 seconds of the server time on the receipt.
1137	ApplVerID	Specifies the application version specifically for this message. ***SUPPORT FOR OTHER APPLICATION VERSIONS MUST BE REQUESTED BEFOREHAND***	String(enum) Must be a valid application version as specified here. **FIX50SP2<9> is the only option initially.**

Obs: All messages must have a SendingTime value within 5 minutes of the server time in UTC or they will be rejected. The timestamp precision should be Millisecond.

Message Trailer

Kalshi uses the standard FIX trailer, as specified here in the Standard Trailer section of the fix spec.

Tag Number	Name	Description	Data Type / Expected Value
10	CheckSum ***Required**	Standard FIX error correction mechanism.	String (3 chars) Value should be calculated by summing the ASCII

		Will be validated on the server. ***MUST BE THE LAST FIELD IN THE MESSAGE***	value of all characters in the message, except for those of the checksum field itself, and then performing modulo 256 over the resulting summation. This is typically implemented automatically on FIX gateway / processors.

Throughout this document we'll be specifying new operations only by defining new values for the MsgType tag, this will be denoted by saying:

OperationX(35=Y) where OperationX is the name of the operation which is defined by a message with tag 35 (MsgType = Y).

Session Layer Messages

The procedure of establishing a session for both the order entry and drop copy session follows the **standards of the FIX protocol.** That means the initiator should send a Logon(35=A) message and the acceptor should:

- Answer with another Logon(35=A) message acknowledging a successful logon.
- Or answer with a Logout(35=5) message in case of failures specifying the reason in the Text(58) field.

Logon (35=A)

Tag Number	Name	Description	Data Type / Expected Value
Header			
98	EncryptMethod ***Required**	Method of encryption in the application layer. Not used by default unless there are specific security considerations that should be communicated out-of-band.	Int (enum) None<0>
96	RawData ***Required**	Client logon message signature, which is a string formed by credentials + a few message fields. Details on how the signature is composed are provided below.	String
1137	DefaultApplVerID ***Required**	Specifies the application version to be used in this session by default. ***SUPPORT FOR OTHER APPLICATION VERSIONS MUST BE REQUESTED BEFOREHAND***	String Must be a valid application version as specified here. **FIX50SP2<9> is the only option initially.**
8013	CancelOrdersOnDisconnect	Whether or not orders for the user will be canceled if there is any kind of disconnection. ***THIS ALSO INCLUDES GRACEFUL LOGOUTS***	Boolean **Defaults to "N".**
20126	ListenerSession	Whether or not this session is only used for listening to execution reports. This is only supported on **KalshiNR** and **SkipPendingExecReports=Y**	Boolean **Defaults to "N".**
21003	SkipPendingExecReports	Whether or not to skip the emission of PENDING_{NEW|REPLACE|CANCEL} exec reports.	Boolean **Defaults to "N".**
Trailer			

The Logon message sent by the client must be signed for security. The prehash string is the following fields joined by the FIX field separator (SOH = ASCII code 1):

PreHashString = SendingTime + SOH + MsgType + SOH + MsgSeqNum + SOH + SenderCompID + SOH + TargetCompID.

There is no trailing separator. The RawData field should be a base64 encoding of the PSS RSA signature using the private key (kalshi-fix.key) for your FIX Api Key.

The procedure for generating the signature that should be used in the RawData<96> tag of the Logon message is:

The Python3 snippet below illustrates the process:

```
from base64 import b64encode, b64decode
from Cryptodome.Signature import pss
from Cryptodome.Hash import SHA256
from Cryptodome.PublicKey import RSA

private_key=open('kalshi-fix.key').read()
private_key_bytes=bytes(private_key, 'utf-8')
private_key = RSA.import_key(private_key_bytes)

# Current timestamp in milliseconds. when the fix message is sent.
sending_time = "20230809-05:28:18.035"
# Fixed as only the Logon message requires signature.
msg_type = "A"
# Fixed as logon message is always expected to have MsgSeqNum=1.
msg_seq_num = "1"
# REPLACE the string below with your SenderCompID.
sender_comp_id = "831bca15-ecf5-428e-979f-25805a383352"
# TargetCompID is always Kalshi.
target_comp_id = "Kalshi"

msg_string = chr(1).join([
  sending_time, msg_type, msg_seq_num,
  sender_comp_id, target_comp_id
])
msg_bytes = bytes(msg_string, 'utf-8')
msg_hash = SHA256.new(msg_bytes)
msg_hash_bytes = pss.new(private_key).sign(msg_hash)
signature_base64_bytes = b64encode(msg_hash_bytes)
# Convert bytes to a string encoding that supports ASCII. Utf-8 works.
raw_data_value = signature_base64_bytes.decode('utf-8')

print(raw_data_value)
```

The raw_data value should look like this:

“oPnbQouQgH8dIBOk8ThSpspGV04+L2fpvWjk/zwJ1WLNefG6/v61R+msNesuDbH1zmcZtz9UXegrmtTjuX78nQY1qYyz63sqLceD4ZyyHAA2xrF0sEpeXrFwpl3U/Alky072NQZiDlTlyGUa33L93wMRM8KtFSDjIlipl21X+ml4y44SqCcUfrdo3Q3rVMAxaoy98oHjBCooS6aKPW37sX78zOVK4UKWDDLfVPVzqn1Y8SFVLbg8UlnVhixEjv62PVvQ0oU5MjkLNVAviwGpvDgFFIP5qOCaxMlHYzW/6HPQijC2x4ohxboXtYGNC5/lsv1umlfPRSjxQrmeFPhlzA==”

If the logon is accepted, a logon ack msg is received, the initiator can start the application messages transmission initiating the *exchange messages step* or commonly known as *trading session*.

Maintaining the Trading Session

If during the trading session a message with an empty value for the MsgSeqNum(34) tag arrives, the acceptor will terminate the FIX connection by sending a Logout(35=5) message with the Text(58) field describing the missing field.

As specified by FIX, receiving **a message with MsgSeqNum(34) lower than the expected NextSeqNumIn is not an expected scenario, this is a serious failure scenario. It means one side received what the other didn’t send.** In that case the connection should be terminated.

On the other hand receiving a message with MsgSeqNum(34) higher than the expected NextNumIn just means that there was a message sent by the other peer that was not received, this is a recoverable scenario, **but we are not implementing this feature on this FIX api yet**.

So if any message arrives in one of the ends with an unexpected MsgSeqNum(34) we suggest the following procedure for reconnecting:

We suggest the following procedure for re-establishing it:

1. Out-of-band communication of the issue providing the logs.
2. The initiator should check the current state of his orders using an alternative channel. Like the REST API for example, or even the UI. In case cancel on disconnect was enabled it is expected that all the orders would be canceled, but it is important to double check.
3. The initiator can establish a new session which will reset sequence numbers.

ResendRequest (35=2)

***Only available on Order Entry with Retransmission session**

Used by the initiator to request retransmission of messages. **We will only support a limit of 2000 messages per request, so you would have to split up the request into chunks of 2000 if required. There is also a time limit of 3 hours of lookback window for messages.**

Tag Number	Name	Description	Data Type
Header			
7	BeginSeqNo	Lower bound (inclusive) of FIX sequence number to resend	Integer
16	EndSeqNo	Upper bound (inclusive) of FIX sequence number to resend	Integer
Trailer			

Session Termination

Given that the trading session was successful and the initiator has no more trading messages to send for the day and wants to disconnect, the initiator should send a Logout(35=5) message that should be replied by the acceptor using the same message type with an empty Text(58) field, followed by the termination of the transport layer connection.

Logout (35=5)

Tag Number	Name	Description	Data Type
Header			
58	Text	Description of the error that caused the logout in case there was any.	String
Trailer			

Important observation: Test Request(35=1) and Heartbeat(35=0) messages are implemented exactly as specified by FIX, which means the connection will be terminated if the response for a heartbeat is not received within the interval agreed. Initially we propose a 30 second heartbeat interval, but we can make it configurable on Logon for a smaller value if required.

Error Handling

The Reject(35=5) message should be issued when a message is received, but cannot be properly processed due to a session-level rule violation.

A case when the reject message is appropriate would be the receipt of a message with unexpected data even if the message is well formed and passes all the validations: the initiator sends a perfectly valid message with correct checksum, but with a message type that is not supported on the Kalshi API.

Generation and receipt of a Reject message indicates a serious error that may be the result of faulty logic in either the sending or receiving application. We recommend logging and communicating those errors for further investigation.

Whenever possible, Kalshi will put the best foot forward on providing the cause of the failure described in the Text <58> field (e.g. INVALID DATA - FIELD 35).

Reject (35=3)

Tag Number	Name	Description	Data Type
Header			
45	RefSeqNum ***Required**	Sequence number of the message that was rejected.	Int
58	Text	Human-readable description of the error.	String
371	RefTagID	Tag number of the field which caused the reject.	Int
372	RefMsgType	The MsgType of the FIX message being referenced.	String
373	SessionRejectReason	Code to identify the reason for the rejection as specified on the fix spec.	Int (enum) Valid values

Important note: This message WILL NOT be used in case the error processing the message is related to the application business logic instead of the session layer. In case the error is related to the business logic, we will use the standard messages designed for the specific purpose and in case there is no appropriate message we will use the generic message BusinessMessageReject(35=j).

BusinessMessageReject (35=j)

Tag Number	Name	Description	Data Type

Header			
45	RefSeqNum ***Required**	Sequence number to start replaying execution reports from.	Int
58	Text	Human-readable description of the error.	String
371	RefTagID	Tag number of the field which caused the reject.	Int
372	RefMsgType	The MsgType of the FIX message being referenced.	String
379	BusinessRejectRefID	The value of the business-level "ID" field on the message being rejected. Will be empty when there is no associated ID.	String
380	BusinessRejectReason ***Required**	Code to identify the reason for a Business Message Reject message as specified on the fix spec..	Int (enum) Valid values

Examples

- Initiator sends a new order single message with MsgSeqNum<34> = 1234, containing the undefined tag number 333333.
    - **The Kalshi acceptor will send a Reject(35=3) message** with fields:
        - RefSeqNum<45> = 1234
        - Text<58> = "Undefined tag received on new order single"
        - RefTagID<371> = 333333
        - RefMsgType<372> = New order single <D>
        - SessionRejectReason<373> = Undefined tag <3>
- Initiator sends a new order single message with MsgSeqNum<34> = 2345, containing all the required tags with valid headers and trailer, during trading business hours on a Kalshi market that is open, but the Kalshi exchange is unavailable for trading because of a technical issue.
    - **The Kalshi acceptor will send a Reject(35=3) message** with fields:
        - RefSeqNum<45> = 2345
        - Text<58> = "Kalshi exchange unavailable"
        - RefMsgType<372> = New order single <D>
        - BusinessRejectReason<373> = Application not available <3>
- Initiator sends a valid new single order just like the previous case, but sends it after the exchange closes.
    - **The Kalshi acceptor will send an ExecutionReport(35=8) message** with OrdRejReason<103> = Exchange closed <2> and other metadata fields to recognize the message that was rejected as specified later in the next section.

Trading Overview

In order to introduce the trading session messages for requests and responses it is necessary to understand how Kalshi markets are structured and Kalshi specific terminology. The next session introduces the concepts required.

Kalshi Market Structure & Terminology

Every market on Kalshi consists of two sides, “Yes” and “No.” Market participants can place bids between 1 and 99 cents to purchase contracts on either side. When two participants place offers for opposite sides and together are willing to pay at least $1, they match and produce a trade.

For example:

1. Trader A places a resting order to purchase 1 Yes Contract for 60 cents on a market.
2. Trader B places an order to purchase 1 No Contract for 45 cents on the same market.
3. As these two offers exceed $1, they match. The price of the resting order is given preference, so trader A receives 1 Yes Contract and gives up 60 cents as collateral, and trader B receives 1 No Contract and gives up 40 cents as collateral. As trader A provided liquidity, they’re internally referred to as the “maker” and trader B is referred to as the “taker.”

At the time of “expiration” (discussed in more detail below) the market outcome is determined to be either Yes or No. If the market outcome is Yes, all traders in the market receive $1 for each Yes Contract they hold, and if the outcome is No, all traders receive $1 for each No Contract they hold.

If, before market expiration, a trader purchases both a Yes and No Contract, they are considered to have offset their position, and $1 is paid out to them immediately. Note that Kalshi’s UI mocks the process of “selling” through the purchase of offsetting contracts, but this is functionally equivalent (by *selling* a Yes Contract, a trader is actually *buying* a No Contract).\

Kalshi’s Market “Hierarchy”

Kalshi’s markets are arranged in an organizational hierarchy to simplify user navigation. At the bottom of this hierarchy are the *markets* themselves - each can be thought to represent a yes or no question, for example: Will the Euro/USD exchange rate be between 1.086 and 1.08799 on Jun 26, 2023 at 6pm EDT?. Each market is assigned an associated unique identifier, the “Market Ticker.”

For example, this Euro/USD market would have the ticker **EURUSD-23JUN2618-B1.087**, which communicates market parameters at a glance. Note that parameters are explicitly communicated using additional metadata attached to every market, and the ticker's structure is intended only for convenience, not programmatic parsing.

Markets which have all but one parametrized dimension in common are grouped into *events*. Events can be thought to represent open ended questions, usually focused on a specific instance in time, for example: "What will the Euro/USD exchange rate be on Jun 26, 2023 at 6pm EDT?" which would contain the market above as well as several other markets with different options for the exchange rate value targets. Events also have their own unique identifier, known as the "event ticker." Event tickers are generally (but not always) a prefix of their markets' tickers. For example, **EURUSD-23JUN2618**.

Finally, events which ask the same question about different points in time are grouped into *series*. Each series represents a particular topic, for example "Euro/USD exchange rate daily." Series also have their own unique identifiers in the form of series tickers, like **EURUSD**.

To summarize, a complete hierarchy might look like:

- Series: EURUSD ("Euro/USD exchange rate daily")
    - Event: EURUSD-23JUN2618 ("Euro/USD exchange rate on 06/26/2023 at 6pm")
        - Market: EURUSD-23JUN2618-T1.08600 (below 1.08600)
        - Market: EURUSD-23JUN2618-B1.087 (between 1.08600 and 1.08799)
        - … (4 more options for 0.00200 ranges in between)
        - Market: EURUSD-23JUN2618-B1.095 (between 1.09400 and 1.09599)
        - Market: EURUSD-23JUN2618-T1.09599 (above 1.09600)
    - Event: EURUSD-23JUN2310 ("Euro/USD exchange rate on 06/23/2023 at 6pm")
        - Market: EURUSD-23JUN2310-T1.09000 (below 1.09000)
        - Market: EURUSD-23JUN2310-B1.091 (between 1.09000 and 1.09199)
        - … (4 more options for 0.00200 ranges in between)
        - Market: EURUSD-23JUN2310-B1.099 (between 1.09800 and 1.09999)
        - Market: EURUSD-23JUN2310-T1.09999 (above 1.10000)

This terminology will appear throughout the document below, Kalshi's broader documentation, and the exchange's UI.

If more details and metadata about the market structure are required we suggest exploration of the REST API.

FIX Representation of Common Kalshi Exchange Terms

Kalshi Term	Description	FIX Tag Name <Number>	FIX value or mapping rule
Market ticker	The ticker that identifies the market. Example: NHIGH-23JAN02-66	Symbol <55>	String value. No mapping required.
Side	Side for the contracts the initiator wants to buy. Sells should be implemented by purchasing an offsetting position on the opposite side.	Side <54>	<table><tr><td>Yes</td><td>Buy <1></td></tr><tr><td>No</td><td>Sell <2></td></tr></table>
Quantity	The number of contracts the initiator wants to buy.	OrderQty <38>	Decimal value Only the integer part will be considered.
Price	Price per contract unit in cents. Needs to be a value between 1 and 99.	Price <44>	Decimal value Only the integer part will be considered.

Order Entry Messages

In Kalshi you can buy Yes and No contracts in the same market, but as we said in the “Exchange Structure & Terminology” section:

> A resting offer to purchase Yes contracts at 4 cents could be matched with another participant willing to purchase No contracts for 96 cents. For that reason, the resting offer represents a “bid” for Yes at 4 cents, or equivalently, an “ask” for No at 96 cents.

To avoid redundancy, the FIX API treats all orders as bids or asks for Yes. With this scheme, traders who take on a net No position can equivalently be seen as taking on a “short” position in Yes; but because Kalshi is fully cash-collateralized, taking on this “short” position requires collateral.

Orders are implemented using **New Order Single <D>** messages following the structure below:

New Order Single (35=D)

Used when the initiator wants to submit a new order.

<table>
<tr><th>Tag No.</th><th>Name</th><th>Description</th><th>Data Type / Observations</th></tr>
<tr><td colspan="4">Header</td></tr>
<tr><td>11</td><td>ClOrderID
*Required</td><td>Order identifier sent by the client to guarantee idempotency.</td><td>String (uuid)</td></tr>
<tr><td>38</td><td>OrderQty
*Required</td><td>Number of contracts for the order.</td><td>Decimal value

Only the integer part will be considered.</td></tr>
<tr><td>40</td><td>OrdType
*Required</td><td>All orders on Kalshi are limit orders.

“Market” orders can be imitated through Immediate or Cancel limit buys at 99 cents or limit sells at 1 cent.</td><td>String (enum)

Allowed values:
- Limit <2></td></tr>
<tr><td>54</td><td>Side
*Required</td><td>Indicates whether the initiator wishes to buy or sell Yes.

As discussed above, selling Yes is equivalent to buying No.</td><td>String (enum)
<table><tr><td>Yes</td><td>Buy<1></td></tr><tr><td>No</td><td>Sell<2></td></tr></table></td></tr>
<tr><td>55</td><td>Symbol
*Required</td><td>Market ticker for the contract you are buying. You can only send orders at Market level.</td><td>String

Example:
“EURUSD-23JUN2618-B1.087”</td></tr>
<tr><td>44</td><td>Price</td><td>Price per contract in cents.</td><td>Decimal value</td></tr>
</table>

	***Required**		Only the integer part will be considered. That part needs to be a value between 1 and 99.
59	TimeInForce	Specify the expiration option for the order. **Defaults to Good Till Cancel if not provided.**	String (enum) - Day <0>: 11:59pm ET of day - Good Till Cancel <1> - Immediate or Cancel <3> - Good Till Date <6>
126	ExpireTime	Should be provided if TimeInForce is Good Till Date <6>.	UTCTimestamp value.
453	NoPartyIDs	Specify the number of associated parties for the order. The only value accepted now is 1	Positive integer value. No mapping required.
=>448	PartyID	Specify the identifier of the sub-account that the order will be placed under. **NoPartyIDs (453) needs to be 1, otherwise this field will be ignored.**	String (uuid)
=>452	PartyRole	Specify the role of the sub-account that the order will be placed under **NoPartyIDs (453) needs to be 1, otherwise this field will be ignored.**	String (enum) - Customer Account (24)
1151	SecurityGroup	Specify the group id to be used in order mass cancel by security group requests	String (uuid recommended)
2964	SelfTradePreventionType	Specify the self-trade prevention behavior when the incoming (taker) order crosses a resting (maker) order from the same trader. If unset, the order will take the default behavior of TakerAtCross **Defaults to TakerAtCross if not provided.**	Integer (enum) - TakerAtCross <1>: The remainder of the taker order is canceled - Maker <2>: The resting order is canceled in full
Trailer			

Execution Report (35=8)

Used to notify the initiator about the order receipt and also any events that happen related to the order posteriorly.

Currently, we use this message type to:

1. confirm the receipt of an order
2. confirm the execution of an order (i.e. resting on orderbook)

3. confirm changes to an existing order (i.e. accept cancel and replace requests)
4. relay fill information on working orders
5. reject orders

TagNo	Name	Description	Data Type
Header			
6	AvgPx	Calculated average price of all fills on this order	Decimal value
11	ClOrderID	ClOrderId identifier provided by the initiator on the last message that made any change to the order this execution report refers to.	String (uuid)
14	CumQty ***Required**	Total number of contracts filled in this order so far.	Decimal value
17	ExecID ***Required**	Unique sequenced identifier for this report message.	String
31	LastPx	Price of this (last) fill in cents. **Only provided if ExecType is Trade<F>.**	Decimal value
32	LastQty	Number of contracts bought or sold in this report. **Only provided if ExecType is Trade<F>.**	Decimal value
37	OrderID ***Required**	Unique identifier for the order in the Kalshi exchange. Please use this ID when referencing the order for support.	String (enum)
38	OrderQty ***Required**	Total number of contracts currently in the order. OrderQty = CumQty + LeavesQty.	Decimal value
39	OrdStatus ***Required**	Conveys the current status of the order. This should be read as the status after the event causing this execution report to be sent.	String (enum) - New<0> - Partially filled<1> - Filled<2> - Canceled<4> - Pending cancel<6> - Rejected<8> - Pending New <A> - Expired<C> - Pending Replace <E>
41	OrigClOrdID	ClOrdID(11) of the previous non-rejected order state. **Only provided if ExecType is Replaced <5> or Canceled <4>**	String (uuid)
44	Price	Price per contract in cents.	Decimal value

<table>
<tr><td>54</td><td>Side
*Required</td><td>Yes or No for the original order.</td><td>String (enum)
allowed values:<table><tr><td>Yes</td><td>Buy <1></td></tr><tr><td>No</td><td>Sell <2></td></tr></table></td></tr>
<tr><td>55</td><td>Symbol
*Required</td><td>Market ticker for the order.</td><td>String

Example:
“EURUSD-23JUN2618-B1.087”</td></tr>
<tr><td>58</td><td>Text</td><td>Human-readable description of the result of the execution report.</td><td>String

Example:
- “insufficient funds”
- “self-trade prevention”
- “position limit exceeded”</td></tr>
<tr><td>60</td><td>TransactTime
*Required</td><td>Timestamp for the event that triggered this execution report.</td><td>UTCTimestamp value.</td></tr>
<tr><td>103</td><td>OrdRejReason</td><td>Specifies the rejection reason in case ExecType = Rejected <8></td><td>String (enum)
- Unknown symbol<1>
- Exchange closed<2>
- Order exceeds limit<3>
- Too late to enter<4>
- Duplicate order<6>
- Unsupported order characteristic<11>
- Other<99></td></tr>
<tr><td>126</td><td>ExpireTime</td><td>Specifies the expiration timestamp for the order if TimeInForce is Good Till Date <6>. This will return 11:59pm ET if TimeInForce is Day<1>.</td><td>UTCTimestamp value.</td></tr>
<tr><td>150</td><td>ExecType
*Required</td><td>Conveys the reason why this execution report was sent. New will typically convey the fact that the new order was received and processed.</td><td>String (enum)
- New <0>
- Canceled <4>
- Replaced <5>
- Pending Cancel <6>
- Rejected <8>
- Pending New <A>
- Expired <C>
- Pending Replace <E>
- Trade <F></td></tr>
<tr><td>151</td><td>LeavesQty
*Required</td><td>Remaining number of contracts that remains open for further execution on this order.</td><td>Decimal value</td></tr>
<tr><td>136</td><td>NoMiscFees</td><td>Number of fees involved in this trade.

Only provided if ExecType is Trade<F>.</td><td>Positive integer value.
No mapping required.</td></tr>
</table>

=>137	MiscFeeAmt	Total fees paid for this trade in dollars.	Decimal
=>138	MiscFeeCurr	Currency of miscellaneous fee. Always dollars.	String (enum) - USD
=>139	MiscFeeType	Indicates type of miscellaneous fee.	String (enum) - Exchange Fees<4>
=>891	MiscFeeBasis	Defines the unit for a miscellaneous fee. We are only using absolute as we report the fees for each individual trade all together, we don't split the fees per contract unit.	Int (enum) - Absolute<0>
704	LongQty	Specify the net Yes position post order-execution.. **Only provided if ExecType is Trade<F> and net position is Yes**	Decimal value
705	ShortQty	Specify the net No position post order-execution. **Only provided if ExecType is Trade<F> and net position is No**	Decimal value
453	NoPartyIDs	Specify the number of associated parties for the order. Only provided if there is a associated party	Positive integer value. No mapping required.
=>448	PartyID	Specify the identifier of the sub-account that the order will be placed under.	String (uuid)
=>452	PartyRole	Specify the role of the sub-account that the order will be placed under	String (enum) - Customer Account (24)
1703	NoCollateralAmounts	Specify the number of collateral accounts that have changed as part of the order **Only provided if ExecType is Trade<F>.**	Positive integer value. No mapping required.
=>1704	CurrentCollateralAmount	Delta of the associated collateral type in cents.	Decimal value
=>1706	CollateralType	Indicates the type of collateral account that is being reported.BALANCE means the change in user current balance. PAYOUT means the netting amount if selling the existing position. A trade could have both BALANCE and PAYOUT change at the same time.	String (enum) - BALANCE - PAYOUT
880	TrdMatchID	Unique identifier for the trade in the Kalshi exchange. Only provided if ExecType is Trade<F>.	String (uuid)
1057	AggressorIndicator	Flag indicating whether the trade was a taker or a maker. Only provided if ExecType is Trade<F>.	Boolean value.
2964	SelfTradePreventionType	Specify the self-trade prevention behavior of the order.	Integer (enum) - TakerAtCross <1>: The remainder of the taker order is canceled - Maker <2>: The resting order is

			canceled in full
Trailer			

The value of ExecType will typically be New to convey the fact that a new order has been received and processed.
However, the value of OrdStatus in this initial response may not necessarily be New as the order might have been executed immediately. The initial value of OrdStatus can therefore also be Partially Filled or Filled. It can even be Canceled if the order has time in force values such as Immediate or Cancel and the order could not be executed immediately.

The OrdStatus is also expected to be Canceled in case the execution report is a response for an OrderCancelRequest(35=F).

Order Cancel Request (35=F)

The order cancel request message requests the cancellation of all of the remaining quantity of an existing order.

This request will only be accepted if the order can successfully be pulled back from the exchange floor without executing.

A cancel request is assigned a ClOrdID <11> and is treated as a separate entity. If rejected, the ClOrdID <11> of the cancel request will be sent in the Cancel Reject <3> message, as well as the ClOrdID <11> of the actual order in the OrigClOrdID <41> field. The ClOrdID <11> assigned to the cancel request must be unique amongst the ClOrdID <11> assigned to regular orders and replacement orders.

Tag Number	Name	Description	Data Type
Header			
11	ClOrderID ***Required**	Unique ID for this cancel request as assigned by the initiator.	String (uuid)
37	OrderID	Unique identifier of the order being canceled that was assigned by Kalshi.	String (uuid)
38	OrderQty ***Required**	Must match the total number of contracts for the order being canceled. OrderQty = CumQty + LeavesQty.	Decimal value
41	OrigClOrdID ***Required**	ClOrdID(11) of the previous non-rejected order. If the order was modified it will be the ClOrdID(11) of the last modification request.	String (uuid)

54	Side ***Required**	Must match the side of the order that OrigClOrdID references.	String (enum) Yes: Buy<1> No: Sell<2>
55	Symbol ***Required**	Must match the symbol (Market Ticker) of the order that OrigClOrdID references.	String example: “EURUSD-23JUN2618-B1.087”
Standard Trailer			

An immediate response to this message is required. Fix recommends that an ExecutionRpt with ExecType <150>=Pending Cancel be sent unless the Order Cancel Request can be immediately accepted (ExecutionRpt with ExecType <150>=Canceled) or rejected (Order Cancel Reject <3> message).

Currently Kalshi does synchronous execution for cancels, which means we will not send a report of PendingCancel exec type and just send the Canceled exec type report immediately. We reserve the right to change that in the future.

Order Cancel Reject (35=9)

The OrderCancelReject(35=9) message is issued by the exchange upon receipt of an OrderCancelRequest(35=F) or OrderCancelReplaceRequest(35=G) message which cannot be honored. Requests to decrease quantity are executed only when an outstanding quantity exists. Filled orders cannot be changed or canceled in any way and you should expect an order cancel reject message in that case.

Tag Number	Name	Description	Data Type
Header			
11	ClOrderId ***Required**	Unique ID for the cancel or replace request this rejection refers to.	String (uuid)
37	OrderId ***Required**	Unique identifier of the order as assigned by Kalshi on creation. Provide extra consistency validation if provided.	String (uuid)
39	OrdStatus ***Required**	OrdStatus value after this cancel/replace rejection is applied.	String (enum) - Filled<2>

			- Canceled<4> - Pending cancel<6>
41	OrigClOrdID	ClOrdID(11) of the order which could not be canceled or replaced. This should be the ClOrdID for the previously accepted order in case there was no change operation, or the ClOrdID for the latest change request on the order otherwise.	String (uuid)
58	Text	A message providing more detail about the cancel reject.	String Example: - “exchange paused”
102	CxlRejReason	A code to identify common reasons why the rejection happened.	String (enum) - Too late to cancel<0> - Unknown order<1> - Other<99>
434	CxlRejResponseTo ***Required**	Specifies if this reject is responding to a cancel or replace request.	String (enum) - Cancel <1> - Replace <2>
Standard Trailer			

When rejecting a OrderCancelReplaceRequest(35=G) (or OrderCancelRequest(35=F)) message, the OrderCancelReject(35=9) message should provide the ClOrdID(11) which was specified on the OrderCancelReplaceRequest(35=G) or OrderCancelRequest(35=F) message for identification, and the OrigClOrdID(41) should be that of the last accepted order – except in the case of CxlRejReason(102) = 1 (Unknown Order).

Important note: If the intention is not to completely cancel the order, but rather just to reduce the quantity, the Order Cancel/Replace Request(35=G) should be used instead of Order Cancel Request (35=F).

Order Cancel/Replace Request or Order Modification (35=G)

Should be used when a client wants to change their order, but not cancel the existing order. We support modifying the following fields of the order

- **(Required)** OrderQty
    - If the order was already filled for more than the OrderQty in the Replace Request, the Replace Request will be rejected
    - If the order was filled for the same number of contracts as the OrderQty in the Replace Request, the order will be canceled.
- (Optional) Price
    - If not provided, the order’s price will be kept at its existing price.

Do *not* use this message to cancel the remaining quantity of an outstanding order, use the OrderCancelRequest(35=F) message for this purpose.

Tag Number	Name	Description	Data Type
Header			
11	ClOrdID ***Required**	Unique ID for this replacement request as assigned by the initiator. This identifier will be used in the ClOrdID field of the Cancel Reject message if this request is rejected.	String (uuid)
37	OrderId	Unique identifier of the order being modified that was assigned by Kalshi. Provide extra consistency validation if provided.	String
38	OrderQty ***Required**	Number of contracts you want to reduce your order to.	Int
40	OrdType ***Required**	All orders on Kalshi are limit orders.	String (enum) Expected value: - Limit <2>
41	OrigClOrdID ***Required**	ClOrdID(11) of the previous non-rejected order. If the order was modified it will be the ClOrdID(11) of the last modification request.	String
44	Price	Price per contract in cents. If not provided, the order's price will remain the same as the existing price.	Decimal value Only the integer part will be considered. That part needs to be a value between 1 and 99.
54	Side ***Required**	Must match the side of the order that OrigClOrdID references.	String (enum) Yes: Buy<1> No: Sell<2>
55	Symbol ***Required**	Must match the symbol (Market Ticker) of the order that OrigClOrdID references.	String example: "EURUSD-23JUN2618-B1.087"
Trailer			

An immediate response to this message is required. It is recommended by the FIX protocol that an ExecutionReport(35=8) with ExecType(150) = E (Pending Replace) be sent unless the OrderCancelReplaceRequest(35=G) can be immediately accepted (ExecutionReport(35=8) with ExecType(150) = 5 (Replaced)) or rejected (OrderCancelReject(35=9) message).

Currently Kalshi does synchronous execution for replacements, which means we will be skipping the PendingReplace exec type and just send the Replaced exec type report right away, we reserve the right to change that in the future as we'll probably have in order to support a higher scale.

The OrderCancelReplaceRequest(35=G) message will only be accepted if the order can successfully be pulled back from the exchange (floor) without executing. Requests which cannot be processed will be rejected using the OrderCancelReject(35=9) message. The OrderCancelReject(35=9) message should provide the ClOrdID(11) and OrigClOrdID(41) values which were specified on the OrderCancelReplaceRequest(35=G) message for identification.

Note that while it is necessary for the ClOrdID(11) to change and be unique, **the exchange's OrderID(37) field will not change** as a result of the OrderCancelReplaceRequest(35=G) message.

The protocol supports the chaining of multiple OrderCancelReplaceRequest(35=G) messages. Care should be taken if the order sender wishes to send a OrderCancelReplaceRequest(35=G) message when there is one or more OrderCancelReplaceRequest(35=G) messages which have not been accepted or rejected – in general:

- The order sender should chain client order IDs on an 'optimistic' basis, i.e. set the OrigClOrdID(41) to the last non rejected ClOrdID(11) sent.

- The order receiver should chain client order IDs on a 'pessimistic' basis, i.e. set the OrigClOrdID(41) on ExecutionReport(35=8) messages that convey the receipt or successful application of a OrderCancelReplaceRequest(35=G) and OrderCancelReject(35=9) message to be the last 'accepted' ClOrdID(11).

In the event that the order sender wants to chain order OrderCancelReplaceRequest(35=G) messages rapidly then they should ensure that each OrderCancelReplaceRequest(35=G) message contains the full details of the order as they would now like it to be.

All of the application-level fields in the original order should be retransmitted with the latest values in the OrderCancelReplaceRequest(35=G) message, except the fields that are being changed.

It is important to note again that the only field we allow changes is OrderQty<38> which must be less than the CumQty<14> + LeavesQty<151> in the last execution report received by the initiator.

Kalshi will validate the OrderCancelReplaceRequest(35=G) message to ensure that the client is not requesting a change for any other field; in this case the Kalshi will send an OrderCancelReject(35=9) message with CxlRejReason(102) = 2 (Broker/Exchange Option).

Mass Cancel Order Request (35=q)

Should be used when a client wants to mass cancel their orders. For now, only cancellations for the entire trading session/ FIX connection is supported. If successful, Kalshi will respond with the MassCancelOrderReport(35=r) followed by individual ExecutionReports(35=8) for each of the canceled orders.

Tag Number	Name	Description	Data Type
Header			
11	ClOrdID ***Required**	Unique ID for this request as assigned by the initiator. This identifier will be used in the ClOrdID field of the MassOrderCancelReport message	String (uuid)
530	MassCancelRequestType ***Required**	Specifying order cancellation criteria. For now, only cancellations for the entire trading session is supported	Enum Allowed value: Cancel orders for a trading session <6> Cancel orders for a security group <A>
1151	SecurityGroup	Specify the group id that was used in new order single requests	String (uuid recommended)
Trailer			

Mass Cancel Order Report (35=r)

This is sent in response to the mass cancel order request.

Tag Number	Name	Description	Data Type
Header			
11	ClOrdID ***Required**	Unique ID for this request as assigned by the initiator. This identifier will be used in the ClOrdID field of the MassOrderCancelReport message	String (uuid)
37	OrderID ***Required**	Unique identifier of the mass cancel operation that was assigned by Kalshi.	String (enum)

530	MassCancelRequestType ***Required**	Specifying order cancellation criteria. For now, only cancellations for the entire trading session is supported	Enum Allowed value: Cancel orders for a trading session <6>
531	MassCancelResponse ***Required**	Specifying the action taken by Kalshi handling the Order Mass Cancel Request	Enum Allowed value: Cancel orders for a trading session <6> Cancel request rejected <0>
532	MassCancelRequestType ***Required**	Indicates why Order Mass Cancel Request was rejected	Enum Allowed value: Cancel orders for a trading session <6>
1151	SecurityGroup	Specify the group id if the mass cancel request was for a security group	String (uuid recommended)
Trailer			

Market Settlement

These messages will be available in both the order entry with retransmission (KalshiRT) and post trade session types.

Market Settlement Report (35=UMS)

This is an exchange-side message that emits necessary information that pertains to the settlement (of funds and closed positions) for a singular market.

Tag Number	Name	Description	Data Type / Observations
Header			
20105	MarketSettlementReportID	Unique identifier for this settlement report.	String value
55	Symbol ***Required**	The ticker string that identifies the market. Example: NHIGH-23JAN02-66	String value.
20106	TotNumMarketSettlementReports	The total number of market settlement reports that will be sent in sequence.	Integer
20107	MarketResult	Result of the market when determined	String value
20108=>	NoMarketSettlementPartyIDs	Number of parties involved in the settlement.	Integer
=>20109	MarketSettlementPartyID	Unique identifier for the parties associated with the settlement	UUID value
=>20110	MarketSettlementPartyRole	Identifies the type of party associated with the settlement	Customer Account <24>
=>704	LongQty	Number of YES contracts held by this party on the settlement.	Integer
=>705	ShortQty	Number of NO contracts held by this party on the settlement.	Integer
=>1703 =>	NoCollateralAmountChanges	Number of collateral changes involved in the settlement. Should be only 1 which is the payout balance change.	Positive integer
==>1704	CollateralAmounChange	Delta in the collateral amount in the specified account in cents	Decimal
==>1705	CollateralAmountType	Type of collateral account that is being modified	String (enum) - Balance <1> - Payout <2>

=>136=>	NoMiscFees	Number of fees involved in this settlement. **Currently the settlement fees are zero. So this repeating group will have a single item with zeroed values.**	Positive integer
==>137	MiscFeeAmt	Total fees paid for this settlement in dollars.	Decimal
==>138	MiscFeeCurr	Currency of this fee. Always dollars.	String (enum) - USD
==>139	MiscFeeType	Indicates type of miscellaneous fee.	String (enum) - Exchange fees <4>
==>891	MiscFeeBasis	Defines the unit for a miscellaneous fee. We are only using absolute as we report the fees for each individual trade all together, we don't split the fees per contract unit.	Int (enum) - Absolute<0>
893	LastFragment	Is this the last page of settlement events	Char Y=Yes N=No
Trailer			

RFQ Messages

QuoteRequest (35=R)

Exchange issues a QuoteRequest message after receiving RFQ from the user.

Tag No.	Name	Description	Data Type / Observations
Header			
131	QuoteReqId ***Required**	Unique identifier corresponding to the quote request	String (uuid)
146	NoRelatedSym ***Required**	Number of related symbol groups	Int Always 1

NoRelatedSym Component Block

38	OrderQty ***Required**	Number of contracts	Decimal value
55	Symbol	Market ticker for the contract you are	String

	***Required**	buying.	Example: “EURUSD-23JUN2618-B1.087”
453	NoPartyIDs	Number of party IDs	Int Always 1

NoPartyIDs Component Block (Nested Repeating Group)

448	PartyId	Pseudonymous identifier for requester	String

QuoteRequestReject (35=AG)

The exchange will issue a QuoteRequestReject if the *user* cancels the QuoteRequest.

Market makers do not need to send a QuoteRequestReject when they are ignoring a QuoteRequest.

Tag No.	Name	Description	Data Type / Observations
Header			
58	Text ***Required**	Reason the quote has been cancelled	String
131	QuoteReqId ***Required**	Unique identifier corresponding to the quote request	String (uuid)
658	QuoteRequest RejectReason ***Required**	Reason the quote has been cancelled	Enum (string) <99> OTHER

Quote (35=S)

Market Maker sends quote after receiving an RFQ. If there is an existing Quote for the market maker, the exchange will attempt to cancel and replace it with the new Quote.

Tag No.	Name	Description	Data Type / Observations
Header			
55	Symbol ***Required**	Market ticker for the contract you are buying.	String

			Example: “EURUSD-23JUN2618-B1.087”
117	QuoteId ***Required**	Unique identifier for the quote	String (uuid)
131	QuoteReqId ***Required**	Unique identifier corresponding to the RFQ for reference	String (uuid)
132	BidPx ***Required**	“Yes” price (in cents) for the given symbol.	Decimal value Only the integer part will be considered. That part needs to be a value between 1 and 99. If BidPx is zero, OfferPx must be non-zero (and vise-versa).
133	OfferPx ***Required**	“No” price (in cents) for the given symbol.	Decimal value Only the integer part will be considered. That part needs to be a value between 1 and 99.

QuoteStatusReport (35=AI)

A QuoteStatusReport is sent by the exchange

1. in response to a Quote. The status will be PENDING, if the exchange has processed the quote, or REJECTED, if the exchange has rejected the quote.
2. when the requester accepts the quote. The status will be ACCEPTED. The quoter should reply with QuoteConfirm within 30 seconds if they want to proceed with execution.
3. in response to a QuoteCancel. The status will be CANCELLED

Tag No.	Name	Description	Data Type / Observations
Header			
38	OrderQty	Number of contracts in the quote. Not available if QuoteStatus REJECTED.	Decimal value Only the integer part will be considered.
54	AcceptedSide	The quote side that has been accepted by the requester. Only available when QuoteStatus ACCEPTED.	Enum (string) Allowed value: <1> YES <2> NO
58	Text	Reason for rejection Only available if QuoteStatus REJECTED.	String

117	QuoteId ***Required**	The unique id for the quote.	String (uuid). Empty string if the quote has been rejected.
131	QuoteReqId ***Required**	Unique identifier corresponding to the RFQ for reference	String (uuid).
132	BidPx	“Yes” price (in cents) for the given symbol. Not available if QuoteStatus REJECTED.	Decimal value
133	OfferPx	“No” price (in cents) for the given symbol. Not available if QuoteStatus REJECTED.	Decimal value
297	QuoteStatus ***Required**	The status of the quote.	Enum (string) Allowed value: <0> ACCEPTED <5> REJECTED <10> PENDING <17> CANCELLED

QuoteCancel (35=Z)

Market Maker sends QuoteCancel. Exchange will issue a QuoteStatusReport with status CANCELLED. Exchange will not issue a QuoteCancel when a Quote is cancelled as the result of a previous Quote.

Tag No.	Name	Description	Data Type / Observations
Header			
117	QuoteId ***Required**	The unique quote identifier this confirmation references.	String (uuid)

QuoteCancelStatus (35=U9)

Market Maker sends QuoteCancelStatus in response to a QuoteCancel message.

Tag No.	Name	Description	Data Type / Observations
Header			
117	QuoteId ***Required**	The unique quote identifier this confirmation references.	String (uuid)
298	QuoteCancelStatus	Whether the quote cancel has been accepted by the exchange.	Enum (string)

	***Required**		<0> CANCELED <1> REJECTED
58	RejectReason	If QuoteCancelStatus is REJECTED, this is populated with the reason.	String

QuoteConfirm (35=U7)

This message is sent by the quote issuer to confirm they are willing to proceed with the quote. If the quote is not confirmed within 30 seconds, the quote is voided.

Tag No.	Name	Description	Data Type / Observations
Header			
117	QuoteId ***Required**	The unique quote identifier this confirmation references.	String (uuid)

QuoteConfirmStatus (35=U8)

Sent in response to a QuoteConfirm message.

Tag No.	Name	Description	Data Type / Observations
Header			
117	QuoteId ***Required**	The unique quote identifier this confirmation references.	String (uuid)
297	QuoteConfirmStatus ***Required**	Whether the quote confirmation has been accepted by the exchange.	Enum (string) <0> ACCEPTED <1> REJECTED
58	RejectReason	If QuoteConfirmStatus is REJECTED, this is populated with the reason.	String

Drop Copy Session

EventResendRequest (35=U1)

Clients can send this message to request order events in the specified range be re-sent. Note that since this is in a different drop copy session, re-sent messages will have new FIX sequence numbers.

Rejects (and any other message that does not contain a valid ExecID (such as pending new orders which have an ExecID of “-1;-1”) will not be resent. For now, only the Execution Report (35=8) message type is supported in this session.

The lookback time window that is supported is of the last 3 hours.

Tag Number	Name	Description	Data Type
Header			
21001	BeginExecID ***Required**	Lower bound (inclusive) of execIDs	String
21002	EndExecID	Upper bound (inclusive) of execIds. If not provided, this will auto resolve to the latest execID in the user’s history	String

EventResendComplete (35=U2)

Sent in response to a successful Event Resend Request following the completion of resending the events.

Tag Number	Name	Description	Data Type
Header			
45	RefSeqNum ***Required**	MsgSeqNum (34) of EventResendRequest	int
21003	ResentEventCount ***Required**	Total number of re-sent events	int

EventResendReject (35=U3)

Sent in response to an Event Resend Request if the request cannot be fulfilled.

Tag Number	Name	Description	Data Type
Header			
45	RefSeqNum ***Required**	MsgSeqNum (34) of EventResendRequest	int
21004	EventResendRejectReason ***Required**	Code identifying reject reason: 1 = Too many resend requests 2 = Server error 3 = BeginExecID is too small 4 = EndExecID is too large	int