Bullish FIX Protocol Specification v1.0.5

Table Of Contents

Index Of FIX Message Tables

Index Of FIX Workflow Diagrams

1. About This Document

1.1. Purpose & Scope

This document describes the FIX Protocol interface to the Bullish platform, covering order and trade entry.

The specification provides details of all business workflows and FIX messages based on FIX 4.4 with some elements from 5.0 SP2 as well as a certain level of Bullish-specific customisation.

The specification assumes the audience's familiarity with the FIX 4.4 protocol, as well as basic understanding of common crypto trading workflows.

1.2. Related Documents

Further details on the technical aspects of operation of the Bullish platform can be found on the Github page at https://github.com/bullish-exchange and at our homepage https://bullish.com.

2. Bullish FIX API Overview

2.1. Hours Of Operation

The Bullish API will ordinarily be available 24 x 7 x 365.

The platform may be closed temporarily from time-to-time should this be required for upgrades or maintenance, and Bullish shall notify clients well in advance when required. Please contact support for more information.

2.2. Network Connectivity

Bullish offers a variety of network connectivity options which include AWS private link and GCP private service connect. Please contact your Bullish relationship manager for details and to discuss which is suitable for you.

2.3. Support

For assistance with this FIX specification, or Bullish functionality generally, please visit our support website at https://support.bullish.com or contact your Bullish relationship manager.

2.4. FIX Dictionary

An up to date copy of the FIX dictionary to complement this FIX specification can be found at https://github.com/bullish-exchange/api-docs/blob/master/src/fix-api/fix-dictionary.xml

2.5. Users, Accounts, Authentication & Security

Bullish clients can create and authorise individual traders (or trading systems) to trade on the Bullish platform with specific authorisations via the trading UI (see Account Management – Bullish Help Center for more information on this). In the case of entering or managing orders, the identity of the instructing trader or system should be indicated in *SenderSubId (50)* of the Standard Header.

In order to log into the FIX session, a *Password (554)* is required in the Logon [A] message. This can be generated by the clients institutional representative via the trading UI (https://exchange.bullish.com) once traders have been declared. Ordinarily, this is a long-lived password (i.e. without expiry); it may, however, change as a result of changing traders and/or accounts. In this case, the FIX session should be closed and a new Logon entered with the new password, to ensure that orders can be entered for the new trader/account.

For more information on managing users and accounts within the system, please contact your Bullish relationship manager.

2.6. Order Account Identification (Allocation)

All orders submitted to the Buillish platform must indicate the account on whose behalf they are submitted, using the *Account (1)* FIX field. The platform will use this information to debit/credit the underlying account following each trade.

2.7. Service Overview

The FIX API allows Bullish customers to:

- Submit and cancel orders on the Bullish platform.
- Receive notification of trades.
- Download reference data about instruments on the platform.

Real-time market data for the Bullish exchange is not currently provided over FIX. This can be received by connecting to the Websockets API, details of which are available at https://api.exchange.bullish.com/docs/api/rest/#overview--websockets.

2.8. FIX, REST & Websocket Interoperability

It is possible to trade via both REST and FIX which would result in the following behaviour:

- Order creation requests sent via REST are published to a subscribed private data websocket (order state updates) only.
- Order creation requests sent via FIX are published to both a subscribed private data websocket (order state updates) and the FIX session (execution reports).
- Order cancellations sent via FIX that result in the cancellation of open orders that were originally sent via REST are published to both a subscribed private data websocket (order state updates) and the FIX session (execution reports).

It is strongly recommended by Bullish to avoid trading over both FIX and REST simultaneously. However, if it is required to do so it is recommended to use separate sub-accounts to segregate your FIX and REST trading flows.

2.9. Symbology

Instruments are typically identified within the Bullish platform using a commonly-understood mnemonic in the format of <quoted currency><base currency>. For example, spot prices for Bitcoin quoted in US Dollar would have a *Symbol (55)* value of `BTCUSD`.

A complete list of instruments available for trading on the Bullish exchange (and their symbols) can be retrieved either via the FIX session using the *SecurityListRequest [x]* message, or from the public `markets` REST endpoint.

2.10. Throttling/Rate limits

Each trading account will be subject to rate limiting on the Bullish platform, with the default set to 50 new orders per second. For more information on increasing your rate limits, please reach out to your Bullish relationship manager.

Should this rate be exceeded, the new order will be rejected with a *ExecutionReport [8]* message indicating *ExecType (150) = 8 (rejected)*, *OrderStatus (39) = 8 (rejected)* and *OrdRejReason (103) = 3034 (Rate limit exceeded)*.

There is an additional global rate limit specific to the exchange. It is used to help limit the flow of orders into the exchange. It affects all clients fairly. When the global rate limit is breached the the new order will be rejected with a *ExecutionReport [8]* message indicating *ExecType (150) = 8 (rejected)*, *OrderStatus (39) = 8 (rejected)* and *OrdRejReason (103) = 3035 (Global rate limit exceeded)*.

Note that order cancellation requests are not subject to rate limiting.

2.11. Cancel on Disconnect

Order entry sessions are configured to automatically cancel all open orders on accounts that have been traded, in the event of disconnection. There is currently no configuration to modify this default cancellation behaviour. Any outstanding *ExecutionReport [8]* messages will be returned during session recovery when connectivity is re-established provided sequence numbers are not reset at logon.

Please note that cancel on disconnect behaviour is tied to trading account id. On session disconnect, it will cancel all open orders on that trading account id. This will also cancel any open orders entered from another active session that used the same trading account id.

Example:
Session 1 - Order 1 - trading account id 1234
Session 2 - Order 2 - trading account id 1234
On Session 1 disconnect it will cancel both Order 1 and Order 2 that used trading account id 1234.
Session 2 will receive Unsolicited cancellation for Order 2

2.12. Timestamps

In line with FIX protocol practice, all timestamps used within this API shall be expressed in Universal Coordinated Time (UTC) to millisecond precision, with the format YYYYMMDD-HH:MM:SS.sss

2.13. Identifiers

Client specified identifier values for ClOrdID(11) should be a unique positive integer expressed as a string in the range 1 to 100,000,000,000,000,000.

2.14. Market Parameters

The Bullish platform offers trading in a variety of symbols, each of which may have its own trading parameters which may constrain both the functionality offered (e.g. order types) as well as pricing and quantity values. These parameters can be accessed via the FIX interface using the SecurityListRequest [x] message.

The following table summarises how various trading attributes vary across the platform:

Price and quantity precision	Both price and quantity are decimal numbers, with the precision for each set at the instrument level.
Minimum/maximum price and quantity	These may vary at the instrument level.
Minimum price increments (tick size)	This may vary at the instrument level, but is typically equal to the smallest price that can be expressed with the price precision.
Order types	Three order types are supported by the platform: limit, market and stop limit.
Time In Force (order validity)	All full range of Time In Force conditions (Good Till Cancel, Fill Or Kill and Immediate Or Cancel) are available for all instruments.

2.15. Error & Rejection Codes

Whilst every effort is made to list error and rejection codes in this document, please visit the link below for the most up-to-date list https://github.com/bullish-exchange/api-docs/wiki/Error-&-Rejection-Codes.

3. FIX Component Definitions

3.1. Standard Header & Trailer

The Bullish platform expects standard FIX message header and trailer components on each FIX message sent and received.

Table 1: Standard Header component

Tag	Field Name	Req'd	Type	Comments
8	BeginString	Y	String	FIX.4.4
9	BodyLength	Y	int	Standard FIX message body length
35	MsgType	Y	String	The message type. See relevant section.
49	SenderCompID	Y	String	Agreed CompID of the party sending the message.
56	TargetCompID	Y	String	Agreed CompID of the intended message recipient. 'BULLISH_PROD' for production use. 'BULLISH_SIMNEXT' for testing.
34	MsgSeqNum	Y	int	FIX Message sequence number
50	SenderSubId	C	String	Required when sent from Client into Bullish (non-session layer message only). The identity of the entering trader or system.
57	TargetSubId	N	String	Echoed from the Bullish system to the Client (where present), or 'ADMIN' for administrative messages not intended for a specific user.
43	PossDupFlag	N	Boolean	Whether the message was previously transmitted under the same MsgSeqNum (34). Absence of this field is interpreted as Original Transmission (N).
97	PossResend	N	Boolean	Whether the message was previously transmitted under a different MsgSeqNum (34). Absence of this field is interpreted as Original Transmission (N).
52	SendingTime	Y	UTCTimestamp	Sending time in UTC
122	OrigSendingTime	N	UTCTimestamp	Original message transmission time in UTC where PossDupFlag (43) = Y. If the original time is not available, this should be the same value as SendingTime (52).

Table 2: Standard Trailer component

Tag	Field Name	Req'd	Type	Comments
10	Checksum	Y	String	Standard FIX checksum

4. FIX Session Level Management

4.1. Establishing and Maintaining a FIX Session

The *Logon [A]* message initiates a connection to the Bullish platform. FIX sessions are typically initiated by Bullish participants, and can be made any time while the system is available.

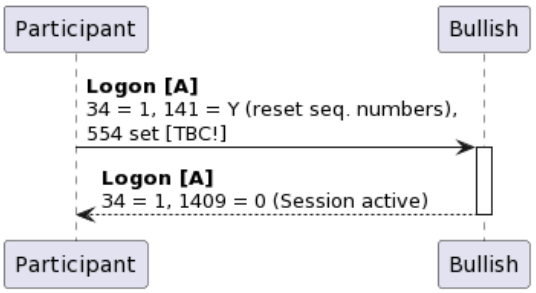
Participants are highly recommended to reset sequence numbers at each logon, which minimises the likelihood of hitting the message cap described in Sequence Number Cap. The option of not resetting sequence numbers is retained here, however, in case the participant wishes to recover messages intra-day in circumstances such as unexpected network disconnects.

Table 3: Logon [A] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = A
98	EncryptMethod	Y	int	Encrypted messages are not supported 0 = NONE
108	HeartBtInt	Y	int	A 30-second interval is required
141	ResetSeqNumFlag	Y	Boolean	Indicates both sides of a FIX session should reset sequence numbers back to 1. We recommend that customers frequently reset sequence numbers at a time convenient for them. This should be 'N' in cases of reconnection after an unexpected disconnect. Y = Yes N = No
554	Password	C	String	Password assigned to the CompID. Required when sent from the client into the Bullish platform.
1409	SessionStatus	C	int	Sent from Bullish to indicate status of Logon attempt. 0 = Session active
< Standard Trailer >		Y		

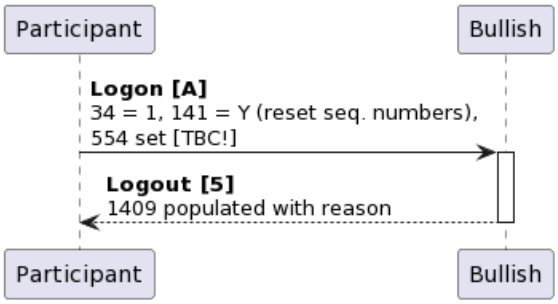
A successful logon attempt shall be acknowledged by the Bullish platform with a *Logon [A]* message indicating *SessionStatus (1409)* = 0 (session active).

Figure 1: Successful logon



Unsuccessful logons shall be rejected using a *Logout [5]* message containing a *SessionStatus (1409)* field to indicate the reason for the rejection.

Figure 2: Unsuccessful logon attempt



Once a FIX session has been successfully established, both FIX engines on either side of the connection should regularly exchange messages to ensure that the connection remains live. The frequency of such a message exchange is determined by the *HeartBtInt (108)* value indicated in the Logon [A] message. The send or receipt of any application message satisfies this requirement to regularly exchange messages. In the event that the session does not send or receive an application message then, then it may instead use the *Heartbeat [0]* message.

Table 4: Heartbeat [0] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = 0 (zero)
112	TestReqID	N	String	Required when the heartbeat is the result of a Test Request [1] message. The value in this field should echo the *TestReqID (112)* received in the Test Request.
< Standard Trailer >		Y		

At any point either counterparty can force the opposing FIX engine to send a Heartbeat [0] message by submitting a *TestRequest [1]* message containing a *TestReqdID (112)* value, which should be echoed in the *Heartbeat [0]* response.

Table 5: TestRequest [1] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = 1
112	TestReqID	Y	String	ID to be returned in the resulting Heartbeat [0] response
< Standard Trailer >		Y		

Figure 3: TestRequest from Bullish triggering a Heartbeat response from participant

4.2. Sequence Number Tracking, Recovery, and Reset

The FIX protocol uses simple, incrementing *MsgSeqNum (34)* values (carried in the Standard Header of each message) to both detect and request retransmission of missed messages.

Each engine should maintain a simple message count of outbound and inbound messages, values which should increment by one for each message sent or received per FIX session. Should a message be received that does not match the expected sequence number given, then it is possible that one or more messages were lost.

In that situation, a *ResendRequest [2]* message may be sent to request retransmission of a specified range of messages identified by their *MsgSeqNum (34)* value.

Table 6: ResendRequest [2] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = 2
7	BeginSeqNo	Y	int	*MsgSeqNum (34)* of first message in the range to be resent (inclusive)
16	EndSeqNo	Y	int	*MsgSeqNum (34)* of the last message in the range to be resent (inclusive), or "0" to request resend all messages after the indicated *BeginSeqNo (7)*
< Standard Trailer >		Y		

The expected response to a *ResendRequest [2]* message is a stream of application messages with *PossDupFlag (43)* set to "Y" (yes) in the header to indicate that this is a potentially duplicative message.

It may be appropriate in some situations for the opposite FIX engine to refuse to replay the message. This is typically the case for administrative messages which either have limited value (e.g., Heartbeat messages) or might cause confusion (e.g., Logon messages). In such circumstances, the opposite FIX engine may respond with a SequenceReset [4] message with *GapFillFlag (123)* set to "Y".

The SequenceReset [4] message can be used in two distinct situations:

- To request that the counterparty adjusts their internal sequence number to the indicated *NewSeqNo (36)* value. In this case, *GapFillFlag (123)* is either not present or set to "N" (no).
- In the message replay situation described above, to replace a FIX message which will not be resent. In this case, *GapFillFlag (123)* should be set to "Y" and the *NewSeqNo (36)* field will contain the next sequence number to be sent.

Table 7: SequenceReset [4] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = 4
123	GapFillFlag	N	Boolean	Indicates that this Sequence Reset is replacing administrative or application messages which will not be resent. Y = Yes N = No
36	NewSeqNo	Y	int	New sequence number
< Standard Trailer >		Y		

4.3. Sequence Number Cap

Since Bullish is a 24 x 7 x 365 venue without regular scheduled downtime, there is a need for a mechanism by which sequence numbers are reset on a regular basis (to avoid them growing uncontrollably).

Sequence number resets require clients to logoff and then login again using the *Logon [A]* message with the *ResetSeqNumFlag (141)* set to 'Y'. This resets the sequence numbers on both sides. Clients are free to perform this action whenever is convenient for their own schedule and set-up.

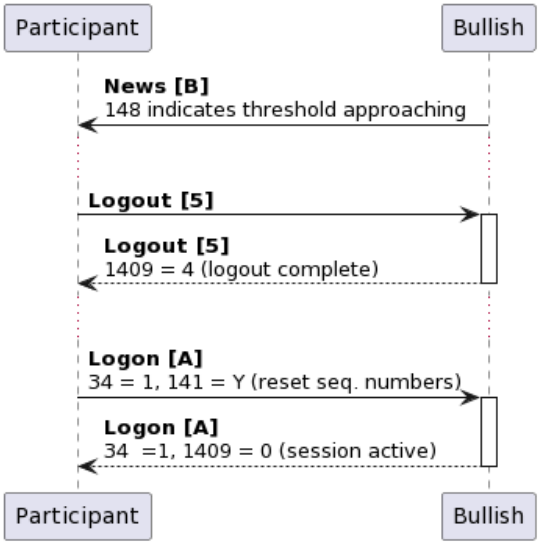
Bullish shall establish a cap to prevent sequence number counter overflow; Bullish shall send an unsolicited *Logout [5]* message should either the inbound or outbound sequence number reach 100,000,000.

To alert clients to an approaching threshold, the Bullish platform shall send an unsolicited *News [B]* message to warn them that they are reaching the maximum threshold, when they reach 1,000 messages less than the threshold. This gives the client the chance to cleanly log out (using a *Logout [5]*) and log back in again (Logon [A]) with the *ResetSeqNumFlag (141)* set to 'Y'.

Table 8: News[B] message indicating an approaching sequence number cap

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = B
148	Headline	Y	String	Text to indicate that the sequence number threshold is approaching.
< Standard Trailer >		Y		

Figure 4: Controlled reset sequence number workflow



4.4. Multiple FIX Sessions

Each FIX connection to the Bullish platform will be assigned a unique *SenderCompID (49)* identifier that should be present in the Standard Header. The client can also request to use their own unique *SenderCompID* value.

Each Bullish client can have up to **20** FIX sessions which can be used to trade across multiple sub-accounts and shared by multiple individual traders (or trading systems).

Clients will be allowed to map multiple senderCompID 's to each institution ID .

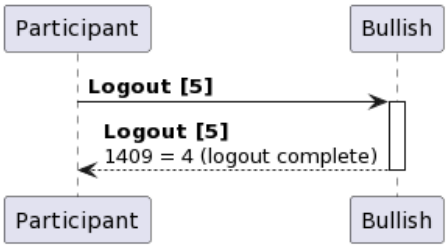
4.5. Terminating a FIX Session

As a 24 x 7 trading platform, Bullish does not have a scheduled "end of day" like traditional markets. Participants are therefore free (and encouraged) to regularly log out and back into their FIX session at a time suitable for their trading activity. By resetting sequence numbers as a result of this action, participants may minimise the likelihood of hitting the Sequence Number Cap.

The *Logout [5]* message initiates or confirms the termination of a FIX session. This would typically be initiated by the Participant, but Bullish may initiate a logout in exceptional circumstances such as planned upgrades or administrative actions.Table 9: Logout [5] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = 5
1409	SessionStatus	C	int	Status of the FIX session. Sent by Bullish server only. 4 = Session Logout Complete 5 = Invalid username or password 6 = Account locked 7 = Logons are not allowed at this time 8 = Password expired 101 = Logout as a result of sequence number threshold 102 = Closed due to platform halt See our Github page for an up-to-date list.
58	Text	N	String	Text specifying reason for the logout.
< Standard Trailer >		Y		

Figure 5: Manual logout workflow



4.6. Handling FIX Errors

In the event that the Bullish platform receives a message that it can not understand or process, it shall respond with a *Reject [3]* message, providing details of the error.

Table 10: Reject [3] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = 3
45	RefSeqNum	Y	int	*MsgSeqNum (34)* of the rejected message.
372	RefMsgType	N	String	*MsgType (35)* of the rejected message.

Tag	Field Name	Req'd	Type	Comments
371	RefTagID	N	String	If a message is rejected due to an issue with a particular field its tag number will be indicated.
373	SessionRejectReason	N	int	Reason for the rejection. See Error & Rejection Codes. 0 = Invalid Tag Number 1 = Required Tag Missing 2 = Tag not defined for this message type 3 = Undefined tag 4 = Tag specified without a value 5 = Value is incorrect (out of range) for this tag 6 = Incorrect data format for value 8 = Signature problem 9 = CompID problem 10 = SendingTime Accuracy Problem 11 = Invalid MsgType 13 = Tag appears more than once 14 = Tag specified out of required order 15 = Repeating group fields out of order 16 = Incorrect NumInGroup count for repeating group 99 = Other
58	Text	N	String	Text specifying the reason for the rejection.
< Standard Trailer >		Y		

5. Order Management

5.1. Workflows

In this section, we shall present a series of order management workflows and concepts at a high level, with references to the relevant FIX messages in subsequent sub-sections. The platform supports three common order types, controlled by *OrdType (40)* which are entered using the *NewOrderSingle [D]* message.

5.1.1. Limit Order (OrdType = 2)

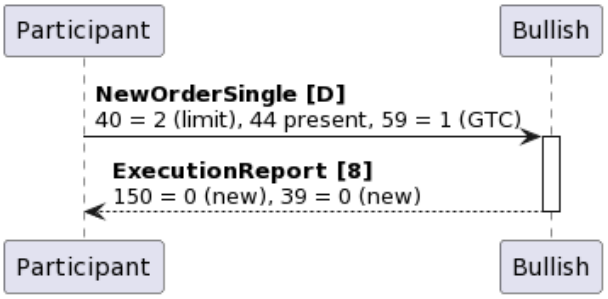
A Limit order is a priced order which is considered for either immediate execution (if a suitable counter order already exists in the market) or can remain in the order book until expiry.

Limit orders can be specified as having one of three possible *TimeInForce (59)* values:

- Good Till Cancel (GTC). The order will remain in the order book until either fully filled or cancelled.
- Fill Or Kill (FOK). The order may either be either immediately and fully executed (possibly against multiple orders) or the full quantity will expire immediately. It is not possible to partially fill an FOK order.
- Immediate Or Cancel (IOC). The order will be immediately filled as much as possible against existing orders, and any remainder will be expired immediately.

Successfully-entered limit orders will be acknowledged by an *ExecutionReport [8]* with *OrdStatus (39)* = 0 (new). If the order receives one or more (immediate) fills, then additional *ExecutionReport [8]* messages will be sent with *OrdStatus (39)* = 1 (partially filled) or 2 (fully filled). If the order carries an FOK or IOC *TimeInForce (59)* condition, then the remaining balance will be immediately expired with a final *ExecutionReport [8]* indicating *OrdStatus (39)* = C (expired).

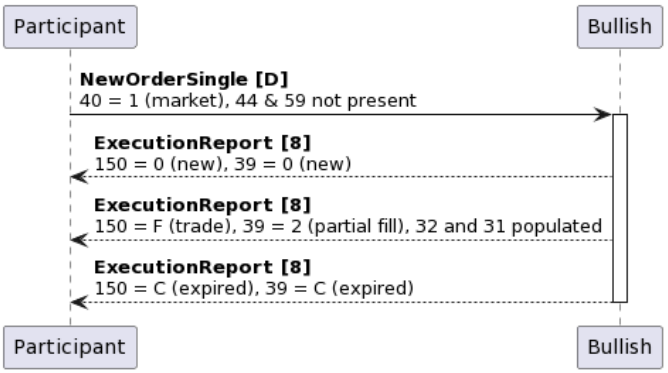
Figure 6: Successful entry and acknowledgement of GTC limit order (without immediate fill)



5.1.2. Market Order (OrdType = 1)

A Market order is an unpriced order type which is considered for immediate execution against other (limit) orders already in the market, with any remainder immediately expiring.

Figure 7: Successful entry of market order which receives a partial fill and the remainder expires



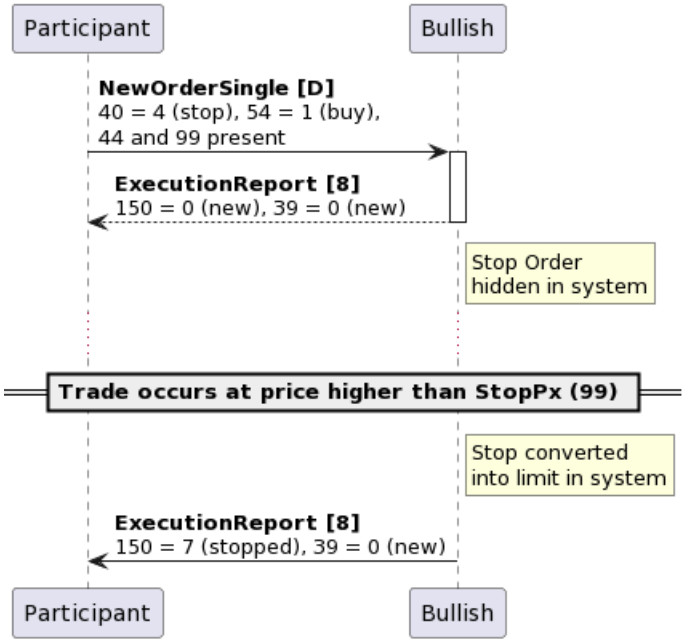
5.1.3. Stop Limit Order (OrdType = 4)

A Stop Limit order is a long-lived order which remains hidden within the platform until the point at which a stop price is triggered. Once a trade occurs in the relevant instrument at a price above the indicated *StopPx (99)* for buy orders, or below the *StopPx (99)* for sell orders, then a limit order is placed into the order book at the indicated *Price (44)*.

The conversion of the Stop Order into a Limit Order is notified to the order originator using an *ExecutionReport [8]* with *ExecType (150)* = 7 (stopped). Once conversion has happened, the order behaves as a normal limit order and can be executed or cancelled as normal.

Since Stop Orders are long-lived orders, the only suitable *TimeInForce (59)* value for them is 1 (GTC).

Figure 8: Successful entry and triggering of a buy Stop Limit order



5.1.4. Margin Trading

The Bullish platform allows institutional users with prior approval to trade on margin, using funds borrowed from other institutional users (peer-to-peer). The system manages the automatic borrowing of assets required to satisfy orders placed, the repayment of those borrowed assets, as well as monitoring the leverage of borrowed assets.

Funds placed in an account can be used as collateral against loans to fulfil margin orders (requiring borrowing) entered into the platform. Each account will be allocated a "Maximum Initial Leverage", governing the maximum amount that can be initially borrowed against collateral (calculated in USD). The system monitors leverage as trading continues and asset prices change. Should leverage become too high, then the system may automatically restrict further borrowing, cancel orders and/or liquidate positions in order to reduce leverage and protect lenders. This liquidation process is described later in this section.

The platform manages both the allocation of loaned assets from other participants, the repayment of those margin loans, as well as the collection and payment of margin interest between the parties. For information on how to loan assets to the margin facility (and therefore earn interest on your assets), please contact Bullish. Note that the FIX API does not currently support the workflows to query margin or loan status or to create loan offers. Any trades arising as a result of the automatic creation of liquidation orders within the Bullish platform will, however, be communicated via FIX.

In case the client has multiple active FIX sessions, the liquidation trades will be sent to all these sessions that are tied to the specific trading account

Orders which require a margin loan to complete them (i.e. the entering user does not have sufficient funds to complete the trade) are entered by setting *CashMargin (544)* = 2 (Margin Open) in the New Order Single [D] message.

Such orders are validated upon entry to confirm that both:

1. sufficient collateral has been posted (up to Maximum Initial Leverage) and,
2. that the pool of assets available to lend can accommodate the request.

If either of these two conditions is not met, then the order will be immediately rejected using an ExecutionReport [8] message with *OrdRejectReason (103)* equal to 2015 (Exceeded max open margin orders) or 3003 (Borrowing is unavailable) respectively.

When a margin order is accepted, the platform will automatically calculate the **additional** amount required to borrow as a result of accepting the order. This incremental borrow quantity is then reserved for the order and the user begins paying interest on the loan immediately. The amount borrowed is communicated via *BorrowedQty (8004)* in the ExecutionReport [8].

Figure 9: Unsuccessful entry of a margin order to buy BTCUSD due to a lack of USD assets

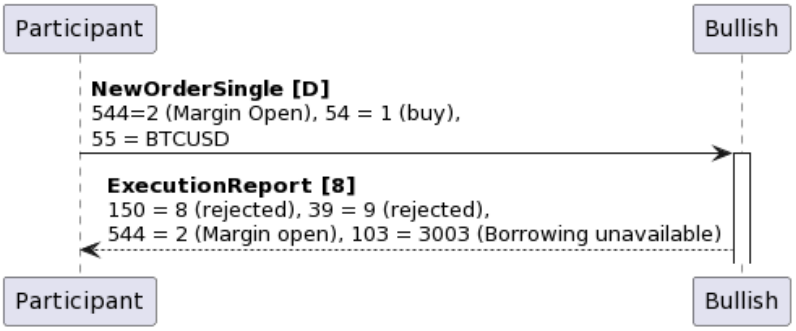
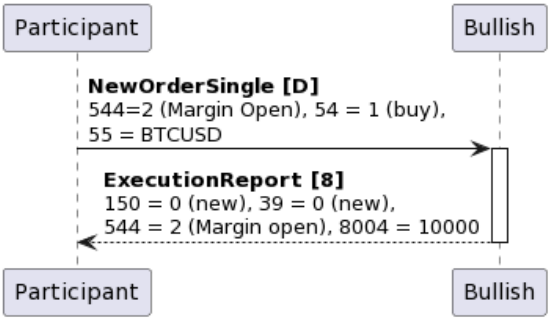


Figure 10: Successful margin order to buy BTCUSD which borrows USD 10,000



Once margin orders have been accepted by the platform, the leverage will be monitored as market conditions change to yield an overall health status. A change in leverage health can impact the range of actions that can be performed by the user as indicated in the table below.

Figure 11: Impact of leverage health on order management

Health Indicator	Healthy	Healthy	Monitor	Caution	Danger	Critical	Suspended
Borrow?	✅	✅	❌	❌	❌	❌	❌
Leverage range	No leverage (1x)	> 1x and < 3x	≥ 3x and < 5x	≥ 5x and < 6x	≥ 6x and < 12x	≥ 12x and <30x	≥ 30x
Trade to increase leverage?	✅	✅	❌	❌	❌	❌	❌
Trade to decrease leverage?	✅	✅	✅	✅	❌	❌	❌
Cancel orders?	✅	✅	✅	✅	❌	❌	❌
Withdraw?	✅	✅	❌	❌	❌	❌	❌
Create AMM Instruction?	✅	❌	❌	❌	❌	❌	❌
Transfer out?	✅	✅	❌	❌	❌	❌	❌

IMPORTANT: this table may change from time to time, please check this knowledge article for the latest description of how leverage impacts the set of actions that can be performed, and how leverage is reduced though liquidation.

Leverage health status can be queried through the Bullish user interface or private Websocket connection.

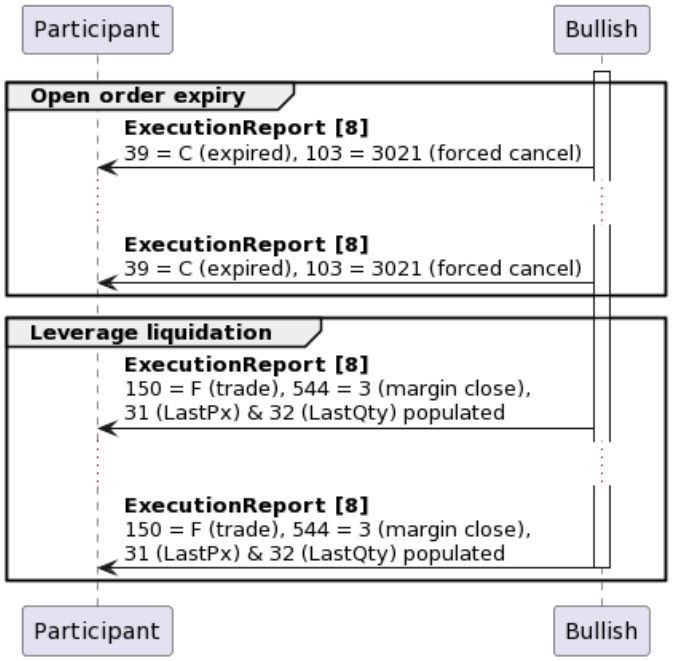
From a FIX order-management perspective, the key leverage health transitions are:

- Healthy => Monitor. Any order which would further **increase** leverage is rejected at the time of order entry with an ExecutionReport [8] indicating *OrderRejStatus (103)* = 3042 (Insufficient loaned balance).
- Caution => Danger. The system will automatically expire all open margin orders by sending an ExecutionReport [8] with *OrderStatus (39)* = C (Expired), and *OrderRejReason (103)* = 3020 (Unsolicited cancel).

Users with poor levels of leverage health can improve their standing by depositing more collateral or trading out of the loans. Should the user fail to take action to improve leverage health, then the Bullish Liquidation Engine will step in to automatically reduce leverage either partially (Danger health level) or fully (Critical health level).

Where the Liquidation Engine submits instructions to reduce leverage, users will receive one or more unsolicited ExecutionReport [8] messages, each indicating *ExecType (150)* = F (Trade) and *CashMargin (544)* = 3 (Margin close) and the details of the trade. These messages will be sent to all active FIX sessions of the users. **Note that since the trade relates to an internal order that was automatically generated by the Bullish Liquidation Engine (as opposed to submitted by the user), the ClOrdID (11) value will NOT appear in these messages as it would not match any reference known by the user's software.**

Figure 12: A leverage health transition from Caution to Danger triggers unsolicited order expiry



5.2. New Order Single (D)

The New Order Single [D] message is used by the client to send orders to the Bullish platform.

Table 11: New Order Single [D] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >		Y		35 = D
11	ClOrdID	Y	String	Unique identifier of the order as assigned by the Bullish customer. Firms submitting multi-day (Good Till Cancel) orders should ensure uniqueness across days.
1	Account	Y	String	Account on which the order is being placed.
544	CashMargin	N	char	Whether the order involves a margin transaction. Where not specified, orders are cash (no margin). 1 = Cash (no margin) 2 = Margin open (borrow to complete the transaction if necessary)
55	Symbol	Y	String	Ticker symbol of the instrument to be traded. See symbology.
54	Side	Y	int	1 = Buy 2 = Sell
60	TransactTime	Y	UTCTimestamp	The time that this order request was initiated/released in UTC
38	OrderQty	Y	Qty	Order quantity expressed as a decimal and with a precision acceptable to the instrument's quantity precision and within the minimum / maximum quantity range for the instrument.
40	OrdType	Y	char	Order type. NOTE: not all order types are accepted for all instruments. 1 = Market 2 = Limit 4 = Stop Limit
44	Price	N	Price	Price of the limit order, or resulting limit order in the case of a triggered Stop Limit order. Required for Order Types of 2 (limit) or 4 (stop limit), and should not be present for 1 (market). NOTE: Price will be validated against the instrument's tick size, and must be within any minimum/maximum price range indicated for the instrument.
99	StopPx	N	Price	Required for Order Type of 4 (stop limit), and should be

Tag	Field Name	Req'd	Type	Comments
				absent in all other cases. NOTE: This price will be validated against the instrument's tick size, and must be within any minimum/maximum price range indicated for the instrument.
59	TimeInForce	N	char	The time validity for the order. Defaults to 1 (GTC) if not present. Ignored for Market Orders. 1 = Good-Till-Cancel (GTC) 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK)
< Standard Trailer >		Y		

5.3. Execution Report (8)

The Bullish platform shall use this message to communicate various order state transitions to Bullish customers, as well as details of completed trades.

Table 12: ExecutionReport [8] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = 8
37	OrderID	Y	String	Unique order reference as assigned by Bullish.
11	ClOrdID	N	String	The last-known *OrderID* as assigned by the originator. Not present for trades arising from orders generated by automated position liquidation. Not present for execution reports from orders entered via REST excluding the *handle* field which were cancelled via FIX.
1	Account	Y	String	Account echoed from order
544	CashMargin	N	char	Whether the order involves a margin transaction. Where not specified, orders are cash (no margin). 1 = Cash (no margin) 2 = Margin open 3 = Margin close
17	ExecID	N	String	Unique reference for the trade (in the case of fills)
1369	MassActionReportID	N	String	Sent in the case of order cancellation as a result of an OrderMassCancelRequest. A unique reference for the action as allocated by Bullish and contained

Tag	Field Name	Req'd	Type	Comments
				in the OrderMassCancelReport.
150	ExecType	Y	char	The purpose of this execution report 0 = New 4 = Cancelled 6 = Pending Cancel 7 = Stopped 8 = Rejected C = Expired D = Restated F = Trade
378	ExecRestatementReason	N	char	The purpose of this execution report 5 = Partial decline of OrderQty<38>
39	OrdStatus	Y	char	The status of the order after the action reported by this ExecutionReport has completed 0 = New 1 = Partially filled 2 = Fully filled 4 = Cancelled 6 = Pending Cancel 7 = Stopped 8 = Rejected C = Expired
103	OrdRejReason	N	int	Reason for order rejection where *ExecType (150)* = 8 (rejected). See Error & Rejection Codes. 1 = Unknown symbol 5 = Unknown order 6 = Duplicate order 13 = Incorrect quantity 15 = Invalid account 18 = Invalid price increment 99 = Other 2013 = Invalid order type 2015 = Exceeded max open margin orders 2021 = Exceeded max open spot orders 3001 = Insufficient account balance 3003 = Borrowing is unavailable 3020 = Unsolicited cancel 3021 = Forced cancel 3031 = Price is out of range 3032 = Order is either closed or rejected 3033 = Leverage increase not permitted 3034 = Order entry throttle reached
58	Text	N	String	Describes why the order is in a specific state
55	Symbol	Y	String	Symbol for order

Tag	Field Name	Req'd	Type	Comments
54	Side	Y	int	Echoed from the order 1 = Buy 2 = Sell
44	Price	N	Price	Echoed from the order (if present)
99	StopPx	N	Price	Echoed from the order (if present)
32	LastQty	N	Qty	Last quantity traded (present for fills only)
31	LastPx	N	Price	Trade price (present for fills only)
6	AvgPx	N	Price	Volume-weighted average of the volume executed so far against this order.
14	CumQty	Y	Qty	The cumulative quantity executed against this order so far (all fills). Value may be zero.
60	TransactTime	Y	UTCTimestamp	Time of this Execution Report message.
38	OrderQty	Y	Qty	Total order quantity echoed from Order Single.
151	LeavesQty	Y	Qty	The remaining quantity available for further execution. Value may be zero for full fills.
8004	BorrowedQty	N	Qty	Margin orders only. The total amount of currency that has been borrowed in order to accommodate this order. *BorrowedQtyCurr* (8006) indicates the currency that was borrowed.
8006	BorrowedQtyCurr	N	Currency	Margin orders only. The currency borrowed in order to accommodate this order, with the quantity borrowed indicated by *BorrowedQty* (8004). This will be the quote currency for sell orders, and the base currency for buy orders.
40	OrdType	Y	char	Echoed from the order 1 = Market 2 = Limit 4 = Stop Limit
59	TimeInForce	N	int	Echoed from the order or defaulted to GTC for limit orders. Not present for Market Orders. 1 = Good-Till-Cancel (GTC) 3 = Immediate Or Cancel (IOC) 4 = Fill Or Kill (FOK)
136	NoMiscFees	N	NumInGroup	Present for fills. Always 1.
→ 137	MiscFeeAmt	N	Amt	The fee amount.

Tag	Field Name	Req'd	Type	Comments
→ 138	MiscFeeCurr	N	Currency	The currency of the fee amount. This will be the quote currency for sell orders, and the base currency for buy orders.
→ 139	MiscFeeType	N	int	Indicates the type of fee described in this repeating group. 4 = Exchange Fee
1057	AggressorIndicator	N	Boolean	Denotes whether this trade was generated by the order taking liquidity (ie it was the aggressor).
< Standard Trailer >				

5.4. Order Cancellation

The Bullish platform supports order cancellation but does not support order amendment.

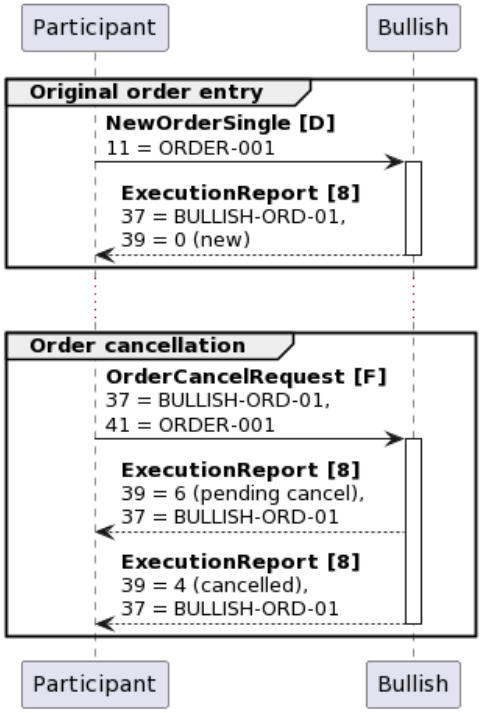
A request to cancel an existing order is made using the *OrderCancelRequest [F]* message, specifying the (Bullish-allocated) *OrderID (37)* reference for the order. A range of order attributes are requested on this cancellation request and are used to validate that the requestor has the same image of the outstanding order as the Bullish system at the time of the cancellation.

Table 13: Order Cancel Request [F] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = F
37	OrderID	N	String	The unique OrderID assigned by Bullish.
41	OrigClOrdID	Y	String	The last-known ClOrdID value for this order
11	ClOrdID	Y	String	The unique reference for this order cancellation request
1	Account	Y	String	Account on which the order is being placed.
55	Symbol	Y	String	Same as in the original order
60	TransactTime	Y	UTCTimestamp	The time of this order cancellation request
< Standard Trailer >				

Since order cancellation requires a check within the matching engine to ensure that the order has not been executed whilst the cancellation request was in flight. For this reason, the expected response to a valid *Order Cancel Request [F]* is an *ExecutionReport [8]* indicating *OrderStatus (39) = 6 (Pending Cancel)*, immediately followed by a second ExecutionReport [8] indicating *OrderStatus (39) = 4 (Cancelled)*.

Figure 13: Successful order entry and subsequent cancellation



In the case that the original order was a margin order, the cancellation of an order may release some borrowed quantity that had been earmarked for that order at the time of order entry (with a corresponding reduction in the user's interest payments). The ExecutionReport [8] message will therefore contain an updated *BorrowedQty (8004)* value, indicating the amount actually borrowed for this order after the cancellation has been completed.

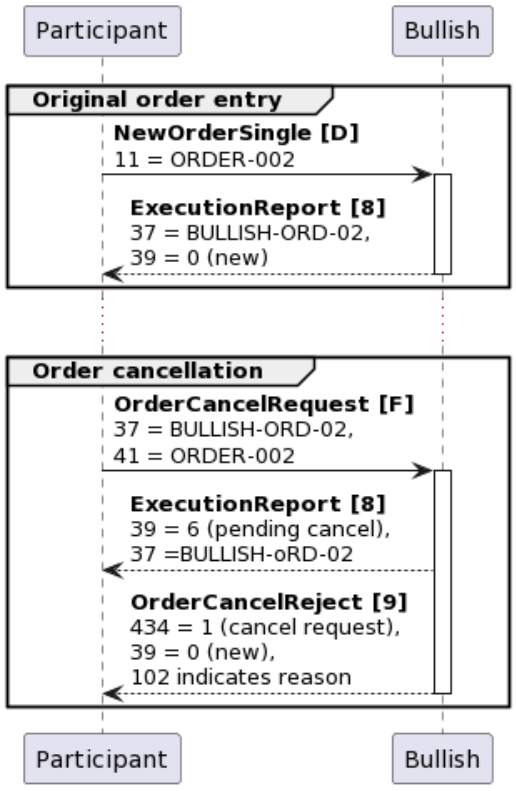
If the cancellation attempt is unsuccessful for some reason, they will receive an *OrderCancelReject [9]* message from the Bullish platform indicating the reason for the rejection.

Table 14: Order Cancel Reject [9] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = 9
37	OrderID	N	String	The unique OrderID assigned by Bullish.
41	OrigClOrdID	Y	String	The last-known ClOrdID value for this order
11	ClOrdID	Y	String	The unique reference for the (rejected) order cancellation request.
39	OrdStatus	Y	Char	Status of the order **after the cancellation rejection**. 0 = New 1 = Partially filled 2 = Fully filled C = Expired

Tag	Field Name	Req'd	Type	Comments
434	CxlRejResponseTo	Y	char	The type of cancellation that is being rejected 1 = Order cancel request
102	CxlRejReason	Y	int	The reason for the rejection. See Error & Rejection Codes. 0 = Too late to cancel 1 = Unknown order 6 = Duplicate ClOrdID (11) received 99 = Other
1328	RejectText	Y	String	Text description of the rejection reason
60	TransactTime	Y	UTCTimestamp	The time of this order cancellation rejection
< Standard Trailer >				

Figure 14: Successful order entry but unsuccessful cancellation



5.5. Mass Order Cancellation

The Bullish platform also supports mass order cancellation in the event that a participant wishes to quickly cancel all orders. This has the same functional outcome as Cancel on Disconnect, except that the FIX session (and network connection) remain connected.

A mass cancellation request is triggered by submitting an *OrderMassCancelRequest [q]* message. The symbol is optional.

Table 15: Order Mass Cancel Request [q] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = q
11	ClOrdID	Y	String	The unique reference for this mass cancellation request
530	MassCancelRequestType	Y	char	The type of cancellation request. 7 = Cancel all orders
1	Account	Y	String	Account on which the order is being placed.
55	Symbol	N	String	Symbol for orders to cancel.
60	TransactTime	Y	UTCTimestamp	The time of this order cancellation request
< Standard Trailer >				

The initial response to the mass order cancellation request is an *OrderMassCancelReport [r]* message.

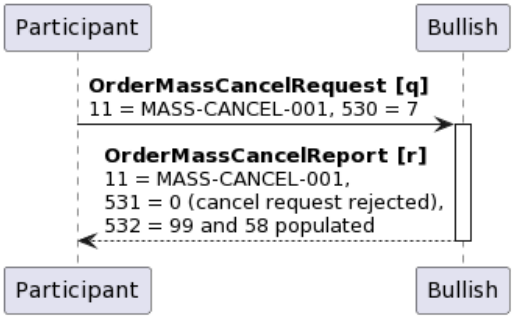
In the case that the mass cancellation request is rejected, this message will indicate *MassCancelResponse (531) = 0 (Cancel Request Rejected)*, with details of the rejection provided in *MassCancelRejectReason (532)* and *Text (58)*.

Table 16: Order Mass Cancel Report [r] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = r
11	ClOrdID	Y	String	The unique reference echoed from the *OrderMassCancelRequest* message.
1369	MassActionReportID	Y	String	Unique reference for the mass cancel action, as allocated by the Bullish platform.
530	MassCancelRequestType	Y	char	The type of cancellation request. 7 = Cancel all orders
531	MassCancelResponse	Y	char	The actions taken as a result of the mass cancellation request. 0 = Cancel Request rejected 7 = Cancel All Orders

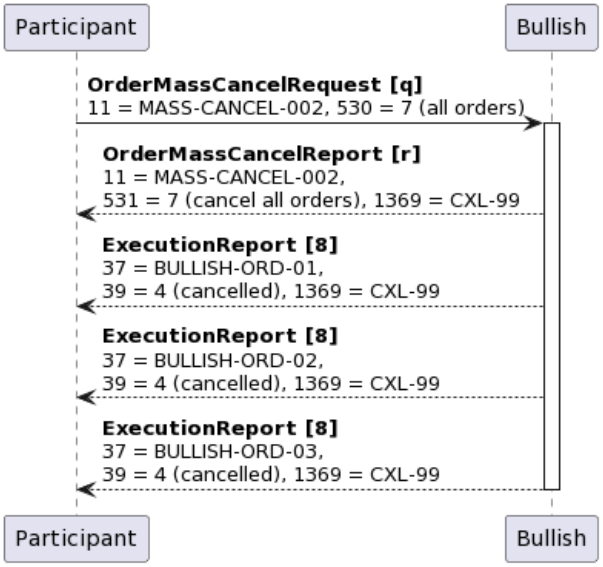
Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = r
11	ClOrdID	Y	String	The unique reference echoed from the *OrderMassCancelRequest* message.
1369	MassActionReportID	Y	String	Unique reference for the mass cancel action, as allocated by the Bullish platform.
532	MassCancelRejectReason	N	int	Sent in the case of mass order cancel rejection. 99 = Other
58	Text	N	String	Sent in the case of mass order cancel rejection to provide a reason for the rejection.
60	TransactTime	Y	UTCTimestamp	The time of this order cancellation report
< Standard Trailer >				

Figure 15: Unsuccessful attempt to mass cancel orders



If the mass order cancel is successful, then the platform will initially respond with an *OrderMassCancelReport [r]* indicating *MassCancelResponse (531) = 7 (Cancel All Orders)*, and then a series of unsolicited *ExecutionReport [8]* messages - one for each order cancelled - which will indicate *OrderStatus (39) = 4 (Cancelled)*, and which carry the unique *MassActionReportID (1369)* identifier to link the action to the mass order cancellation.

Figure 16: Successful mass order cancellation



6. Reference Data

Static reference data related to crypto pairs traded on the Bullish platform can be requested by submitting the *SecurityListRequest [x]* message. This can be used to request details about the market for one or more cryptocurrencies by setting:

- SecurityListRequestType (559) = 0 (single security) and specifying Symbol(55), or
- SecurityListRequestType (559) = 4 (all securities)

Table 17: SecurityListRequest [x] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = x
320	SecurityReqID	Y	String	Unique ID of the request.
559	SecurityListRequestType	Y	int	Type of SecurityList Request being made 0 = Single Symbol 4 = All securities
55	Symbol	C	String	Required where SecurityListRequestType (559) = 0 (Single Symbol)
< Standard Trailer >				

Upon receipt of the *SecurityListRequest [x]* message, the Bullish platform will respond with a single *SecurityList [y]* message containing a list of matching crypto instruments; this will be a single instrument if the *SecurityListRequest [x]* specified a single Instrument, or all instruments on the platform.

Note that the *SecurityList [y]* message is fragmented.

Table 18: SecurityList [y] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = y
320	SecurityReqID	Y	String	The unique ID of the request.
560	SecurityRequestResult	Y	int	Status result of the SecurityListRequest. See Error & Rejection Codes. 0 = Valid request 1 = Invalid or unsupported request 2 = Instrument not found
1607	SecurityRejectReason	N	int	Used to specify a rejection reason when SecurityResponseResult (560) = 1 (Invalid

Tag	Field Name	Req'd	Type	Comments
				or unsupported request).
60	TransactTime	Y	UTCTimestamp	Timestamp that this message was sent.
893	LastFragment	N	Boolean	Indicates if this message in a fragmented response is the final one of the sequence
146	NoRelatedSym	Y	int	Number of symbols indicated in this message
→ 55	Symbol	Y	String	Instrument symbol
→ 1647	NoRelatedInstruments	Y	int	Always 2
→→ 1649	RelatedSymbol	Y	String	The related symbol (either base or quoted currency)
→→ 1652	RelatedSecurityType	Y	String	The type of the related symbol DIGITAL = Cryptocurrency CASH = Fiat currency
→ 969	MinPriceIncrement	Y	float	Minimum price increment for a given instrument (ie the tick size)
→ 8000	MinQtySize	Y	float	Minimum quantity for a given instrument
→ 8001	MaxQtySize	Y	float	Minimum quantity for a given instrument
→ 8002	PricePrecision	Y	int	The precision to which prices can be specified (integer representing the number of digits after the decimal point)
→ 8003	QtyPrecision	Y	int	The precision to which quantities can be specified (integer representing the number of digits after the decimal point)
→ 8005	MarginEnabled	Y	Boolean	Whether margin trading is enabled on this instrument or not.
< Standard Trailer >				

Figure 17: Request reference data for a single instrument

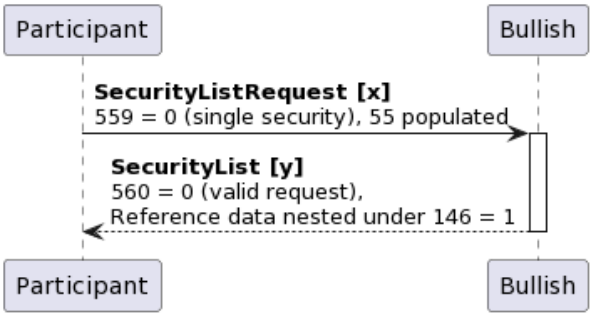


Figure 18: Request reference data for ALL instruments

7. Trading Status

As a 24 x 7 x 365 exchange, the Bullish platform should ordinarily always be available.

From time to time there are, however, periods of platform-wide downtime for scheduled maintenance or updates. These will be communicated to participants in advance (contact Market Operations to subscribe for notifications).

Additionally, there may be reasons for individual instruments to be temporarily removed from trading.

The platform will communicate both of these situations through the *TradingSessionStatus [h]* message which is broadcast from Bullish.

Table 19: TradingSessionStatus [h] message

Tag	Field Name	Req'd	Type	Comments
< Standard Header >				35 = h
336	TradingSessionID	Y	String	The trading session this message relates to. 1 = Day.
325	UnsolicitedIndicator	Y	Boolean	These messages are always unsolicited Y = Unsolicited
340	TradSesStatus	Y	int	Status of the platform or instrument (where indicated) 1 = Halted 2 = Open
1368	TradSesEvent	Y	int	Identified the event causing the state change 0 = Trading resumes 3 = Change of Trading Status
58	Text	N	String	Further detail on the reason for the status change
55	Symbol	N	String	If present, then the change in session status shall relate to this one specific instrument. If omitted, the message should be interpreted as a platform-wide status change.
< Standard Trailer >				