ASX Trade

ASX Trade ITCH Specification

August 2019

Table of Contents

1 Introduction....4
1.1 ASX ITCH....4
1.2 Glimpse....5
1.3 Document Information....6
1.3.1 Format of ITCH example messages....6
1.4 Version History....6
2 ASX ITCH....7
2.1 Architecture....7
2.1.1 Protocol....7
2.2 Data Types....7
2.3 Message Formats....7
2.3.1 Time Messages....7
2.4 Reference Data Messages....8
2.4.1 Order Book Directory....8
2.4.2 Combination Order Book Directory....9
2.4.3 Tick Size Table Entry....11
2.5 Event and State Change Messages....11
2.5.1 System Event Message....11
2.5.2 Order Book State Message....12
2.6 Market by Order Messages....12
2.6.1 Add Order Messages....12
2.6.2 Modify Order Messages....14
2.7 Trade Messages....17
2.8 Auction Messages – Equilibrium Price Update....18
2.9 Building an Order Book View....18
2.10 Building a Trade Ticker....19
3 Glimpse....20
3.1 Architecture....20
3.1.1 Protocol....20
3.2 Data Types....20
3.3 Glimpse Messages....20
3.3.1 End of Snapshot Message....20
4 Appendix 1 – ASX ITCH Scenarios....21
4.1 Normal Orders....21
4.1.1 New order entered, amended and deleted....21
4.1.2 New order entered, fully trades....21
4.1.3 New order entered, partially trades....21
4.1.4 Existing order amended, and fully trades....22
4.1.5 Existing order amended, and partially trades....22
4.1.6 Orders trade in auction....22
4.2 Undisclosed Order, Continuous Trading....23
4.2.1 Undisclosed Order Entered Into Book....23

4.2.2 New Order Fully Trades with Existing Undisclosed Order – 1 23
4.2.3 New Order Fully Trades with Existing Undisclosed Order – 2 24
4.2.4 New Order Fully Trades with Existing Undisclosed Order – 3 24
4.3 Iceberg Order Execution, Continuous Trading 24
4.3.1 New Order Fully Trades with Existing Iceberg Order - 1 24
4.3.2 New Order Fully Trades with Existing Iceberg Order – 2 24
4.3.3 New Order Fully Trades with Existing Iceberg Order – 3 25
4.3.4 New Order Fully Trades with Existing Iceberg Order - 4 25
4.4 Tailor Made Combinations 25
4.4.1 New TMC Order Trades with Existing TMC Order 25
4.4.2 New TMC Order Trades with Existing Orders in the Book 26
4.4.3 TMC Order Trades with an Opposing TMC Order during an Auction 27
4.5 Miscellaneous 28
4.5.1 Standard Instrument Creation 28

5 Appendix 2 – ASX MoldUDP64 Protocol Specification 29

5.1 Overview 29
5.2 Assumptions 29
5.3 Terms 29
5.3.1 Message 29
5.3.2 Session 29
5.4 Downstream Packet 30
5.4.1 Downstream Packet Header 30
5.4.2 Sequence Number 30
5.4.3 Message Count 30
5.4.4 Downstream Packet Message Block 30
5.4.5 Message Length 30
5.4.6 Message Data 30
5.4.7 Heartbeats 31
5.4.8 End of Session 31
5.4.9 Request Packet 31
5.4.10 Sequence Number 31
5.4.11 Requested Message Count 31
5.5 Receiver Example 31

6 Appendix 3 - SoupBinTCP Protocol 33

6.1 Overview 33
6.1.1 SoupBinTCP Logical Packets 33
6.1.2 Protocol Flow 33
6.1.3 Heartbeats 34
6.1.4 End of Session Marker 34
6.1.5 Data Types 34
6.2 SoupBinTCP Packet Types 34
6.2.1 Debug Packet 34
6.2.2 Logical Packets Sent by a SoupBinTCP Server 34
6.2.3 Logical Packets Sent by the SoupBinTCP Client 36

1 Introduction

ASX ITCH is a direct data feed delivered via User Datagram Protocols (UDPs) and Transmission Control Protocols (TCPs). ASX ITCH enables the tracking of the status of an order from the time it is first entered until the time it is either executed or cancelled. Administrative messages can also be received. ASX ITCH is intended for information exchange only. Active trading is supported by the ASX OUCH protocol.

ASX ITCH is a binary protocol for accessing ASX Market Information, delivered via a multicast connection directly from the ASX Trade platform. ASX ITCH has been developed to maximise performance and to meet the requirements of latency sensitive traders.

The ASX ITCH protocol provides:

- Control of the socket
- Multicast stream of order book changes
- Full order book detail for all ASX asset classes - equities, index options, single stock options, interest rate, funds, ETFs and structured products
- Trade data for all ASX order books (ASX TradeMatch, Centre Point and PureMatch)
- Access to all asset classes available in ASX Trade
- Access to security status messages
- Access to basic security data including ISIN code, financial product and tick size
- Internationally recognised and standardised protocol
- Time-stamping from ASX Trade to the nano-second.

ASX offers two test environments for the ASX ITCH service. These environments are independent from each other. The list of instruments on each environment contains different instruments, and where an instrument name is the same, the reference data within it may be different.

1.1 ASX ITCH

ASX ITCH is a direct data feed product that features the following data elements:

- Order level data Market by Order (MBO) with broker ID - The system provides its full order depth using the standard ASX ITCH format. ASX ITCH uses a series of order messages to track the life of a customer order. The ASX ITCH message displays customer ID for non-anonymous instruments.
- Trade messages - ASX ITCH supports trade messages to reflect matches in lit and dark order books, with the exception of trade reports.
- Reference data:
- Order Book Directory messages provide basic security data such as the ISIN code and financial product
- Tick Size Table Entry messages to convey tick sizes for order books.
- Event controls, such as the states of the different order books
- Order Book State message to inform receivers of state changes.

1.2 Glimpse

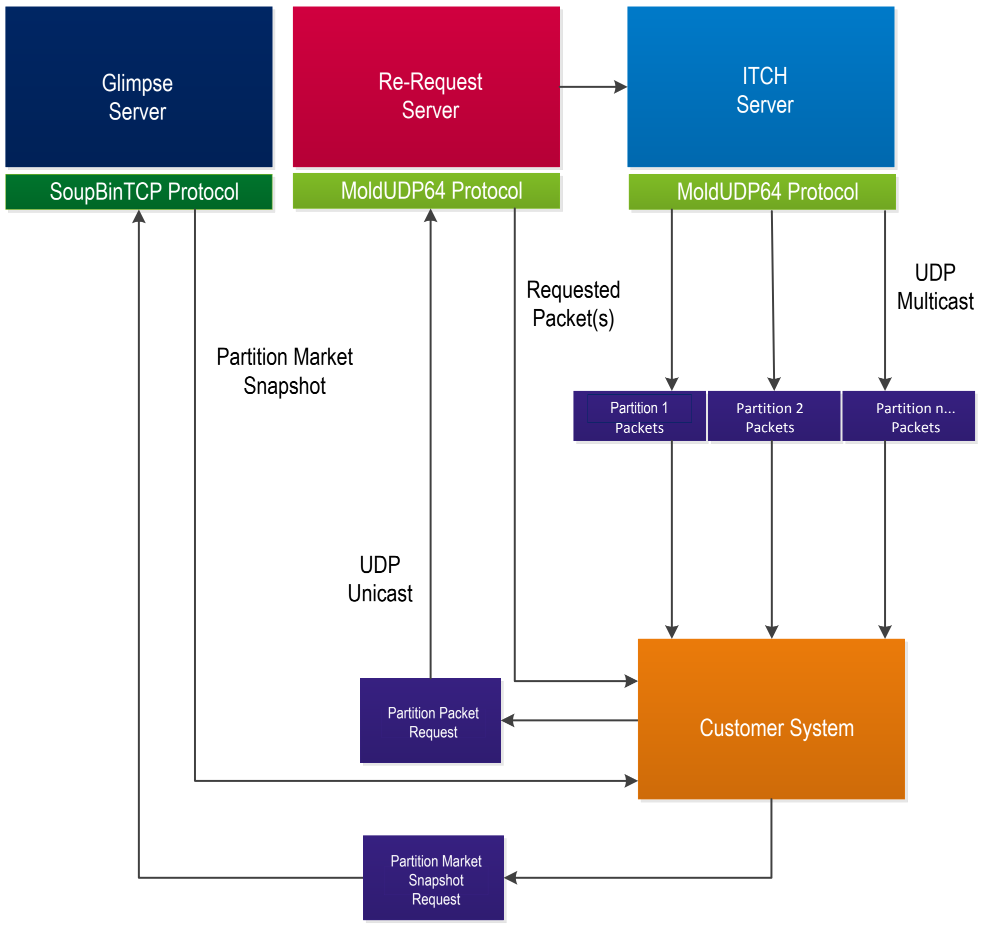
Complementing the ASX ITCH real-time data feed product, Glimpse is a point-to-point data feed connection that provides direct data feed customers with a snapshot of the current state of the order books traded in ASX Trade. Glimpse uses the same message formats as ASX ITCH.

Glimpse can be used to quickly sync up with the ASX ITCH feed. At the end of the Glimpse snapshot a sequence number is provided that can be used to connect and sync up with the real-time ASX ITCH feed.

Glimpse provides the following:

- Basic Reference Data for each order book including intra-day updates up until the time of login
- Current trading state of each order book
- All lit orders for each order book
- An End of Snapshot message providing the ASX ITCH sequence number to use when connecting to the real-time ASX ITCH feed.

The following diagram shows the relationship between the ASX ITCH, Glimpse and Re-Request Server. For more information on the MoldUDP64 protocol, see *5 Appendix 2 – ASX MoldUDP64 Protocol Specification*. For more information on the SoupBinTCP see *Appendix 3 - SoupBinTCP Protocol*.



For ASX ITCH/Glimpse questions contact ASX Customer Technical Support (CTS) team either via email on cts@asx.com.au or phone 1800 663 053 (or on +61 2 9227 0372 from outside Australia).

1.3 Document Information

1.3.1 Format of ITCH example messages

As a binary protocol, ITCH messages are transmitted as a series of bytes and it is up to the receiver to apply the definitions in this manual to interpret the contents of a message. In displaying the examples, the messages in this manual are shown in a translated form. Messages are translated as follows:

- Each field is separated from the next by a comma (there is no separator in the ITCH message)
- Data types are displayed as follows:

  - Numeric: unsigned integers, with the exception of Order ID and Match ID, which are displayed as blocks of 4 bytes in hexadecimal format, separated by colons
  - Price: signed integers
  - Alpha: Left justified, spaces trimmed to the right

1.4 Version History

This document has been revised according to the table below:

Version	Date	Comment
v2.0	Mar 2015	• The Login Rejected Packet, the Packet Type Field Name was changed had the value I changed to J
v2.1	December 2018	• Updated to new ASX branding • Removal of connectivity section – moved to a new document • Addition of missing exchange order type to *Add Order* messages • Addition of information on when the *Order Replace* message is sent. Updated example illustrating a case when sent • Updated the description of ITCH • Updated format of examples, and all examples
v2.2	August 2019	• Updated Introduction • Additional Trade Messages information • Addition of Logout Request Packet • Updated Client Heartbeat Packets • Appendix 1 update

2 ASX ITCH

2.1 Architecture

The ASX ITCH feed is made up of a series of sequenced messages. Each message is variable in length, based on the message type. The messages will be binary encoded using MoldUDP64. The messages that make up the ASX ITCH protocol are delivered using a higher level protocol that takes care of sequencing and delivery guarantees.

2.1.1 Protocol

The ASX ITCH data feed is offered in the following:

Protocol Option	Description
MoldUDP64	MoldUDP64 is a light-weight networking protocol built on top of UDP that provides a mechanism for listeners to detect and re-request missed packets. Each message is explicitly sequence numbered. If a packet loss is detected by the client, it can re-request that packet from the MoldUDP64 rewind server, and it will be resent as a UDP unicast to that client.

2.2 Data Types

All numeric fields are composed of binary encoded numbers.

All alpha fields are left justified and padded on the right with spaces. The alpha fields are composed of non-control ISO 8859-1 (Latin-1) encoded bytes.

Type	Size	Notes
Numeric	1, 2, 4, 8 or 16 bytes	Unsigned big-endian binary encoded numbers. Note: The transport layer, MoldUDP64, uses big-endian for its numeric values.
Alpha	Variable	Left justified and padded on the right with spaces.
Price	4 bytes	Prices are signed big-endian fields. Number of decimals is specified in the Order Book Directory message. Note: The number of decimal places for prices in ASX ITCH will universally be disseminated with 2 decimal places, regardless of the configured number of decimal places for the instrument in ASX Trade.

2.3 Message Formats

The ASX ITCH feed is composed of a series of messages that describe orders added to, removed from and executed on ASX Trade. It also contains messages for basic reference data of the order books as well as state changes and halts.

2.3.1 Time Messages

For bandwidth efficiency, the ASX ITCH timestamp is separated into two parts:

Timestamp Portion	Message Type	Notes
Seconds	Standalone message	Unix time (number of seconds since 1970-01-01 00:00:00 UTC). **Note**: A Timestamp - Second message will be disseminated for every second for which there is at least one message.
Nanoseconds	Field within individual messages	Reflects the number of nanoseconds since the most recent Timestamp - Seconds message that the data message was generated.

2.3.1.1 Seconds Message

This message is sent every second for which at least one ASX ITCH message is being generated. The message contains the number of seconds since the start of 1970-01-01 00:00:00 UTC, also called Unix Time.

Name	Length	Value	Notes
Message Type	1	"T"	Seconds Message.
Second	4	Numeric	Unix time (number of seconds since 1970-01-01 00:00:00 UTC).

2.4 Reference Data Messages

2.4.1 Order Book Directory

At the start of each trading day, Order Book Directory messages are disseminated for all active instruments.

Note:
Intra-day transmissions of this message will occur when new order books (instruments) are added to the system. Updates to existing order books will also be represented by intra-day Order Book Directory messages.

Name	Length	Value	Notes
Message Type	1	"R"	Order Book Directory Message
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	Denotes the primary identifier of an order book (instrument). **Note**: Expired Order Book IDs may be reused for new instruments.
Symbol	32	Alpha	The unique instrument series name (ins_id_s in ASX Trade).
Long Name	32	Alpha	Additional instrument series information. May be blank and may not necessarily be unique across all series (long_ins_id_s in ASX Trade).
ISIN	12	Alpha	ISIN code identifying the instrument.
Financial Product	1	Numeric	Values: 1 = Option 3 = Future 5 = Cash 11 = Standard combination. **Note**: Warrants have a value of 1 (Option).
Trading Currency	3	Alpha	Trading currency
Number of decimals in Price	2	Numeric	This value defines the number of decimals used in price for this order book. **Note**: The number of decimal places for prices in ASX ITCH will universally be disseminated with 2 decimal places, regardless of the configured number of decimal places for the instrument in ASX Trade.
Number of decimals in Nominal Value	2	Numeric	This value defines the number of decimals used in the nominal value for this order book.

Name	Length	Value	Notes
Odd Lot Size	4	Numeric	Indicates the number of securities that represent an odd lot for the order book. **Note**: A value of 0 indicates that this lot type is undefined for the order book.
Round Lot Size	4	Numeric	Indicates the number of shares that represent a round lot for the order book.
Block Lot Size	4	Numeric	Indicates the number of securities that represent a block lot for the order book. **Note**: A value of 0 indicates that this lot type is undefined for the order book.
Nominal Value	8	Numeric	Nominal value

Order Book Directory Example Message

Order book directory for CBA:

```
R,911814500,85603,CBA,CWLTH BANK FPO [CBA],AU000000CBA7,5,AUD,2,0,0,1,0,0
```

2.4.2 Combination Order Book Directory

The Combination Order Book Directory is a specialised directory message used for combinations. It represents both standard combinations defined by ASX, and tailor-made combinations created by customers.

Note:
Intra-day transmissions of this message will occur when new combination order books are added in ASX Trade. This is typically the case for tailor-made combinations. Updates to existing combination order books may also be represented by intra-day Combination Order Book Directory messages.

Name	Length	Value	Notes
Message Type	1	“M”	Combination Order Book Directory Message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	Denotes the primary identifier of an order book (instrument). **Note**: Expired Order Book IDs may be reused for new instruments.
Symbol	32	Alpha	The unique instrument series name (ins_id_s in ASX Trade).
Long Name	32	Alpha	Additional instrument series information. May be blank and may not necessarily be unique across all series (long_ins_id_s in ASX Trade).
ISIN	12	Alpha	ISIN code identifying security.
Financial Product	1	Numeric	Values: 1 = Option 3 = Future 5 = Cash 11 = Standard combination. **Note**: Warrants have a value of 1 (Option)
Trading Currency	3	Alpha	Trading currency

Name	Length	Value	Notes
Number of decimals in Price	2	Numeric	This value defines the number of decimals used in price for this order book. **Note:** The number of decimal places for prices in ASX ITCH will universally be disseminated with 2 decimal places, regardless of the configured number of decimal places for the instrument in ASX Trade.
Number of decimals in Nominal Value	2	Numeric	This value defines the number of decimals used in the nominal value for this order book.
Odd Lot Size	4	Numeric	Indicates the number of securities that represent an odd lot for the order book. **Note**: A value of 0 indicates that this lot type is undefined for the order book.
Round Lot Size	4	Numeric	Indicates the number of shares that represent a round lot for the order book.
Block Lot Size	4	Numeric	Indicates the number of securities that represent a block lot for the order book. **Note**: A value of 0 indicates that this lot type is undefined for the order book.
Nominal Value	8	Numeric	Nominal value
Leg 1, Symbol	32	Alpha	The unique series name of the leg.

Leg 1, Side	1	Alpha	Values:
			B = Buy leg
			C = Sell leg.

Leg 1, Ratio	4	Numeric	Relative numbers of contracts in comparison to other legs.
Leg 2, Symbol	32	Alpha	The unique series name of the leg.

Leg 2, Side	1	Alpha	Values:
			B = Buy leg
			C = Sell leg.

Leg 2, Ratio	4	Numeric	Relative numbers of contracts in comparison to other legs.
Leg 3, Symbol	32	Alpha	The unique series name of the leg.

Leg 3, Side	1	Alpha	Values:
			B = Buy leg
			C = Sell leg
			? = not defined (i.e. no leg 3 present).

Leg 3, Ratio	4	Numeric	Relative numbers of contracts in comparison to other legs.
Leg 4, Symbol	32	Alpha	The unique series name of the leg.

Leg 4, Side	1	Alpha	Values:
			B = Buy leg
			C = Sell leg
			? = not defined (i.e. no leg 4 present).

Leg 4, Ratio	4	Numeric	Relative numbers of contracts in comparison to other legs.

Combination Order Book Directory Example Message

Example of a TMC with two legs.

```
M,546625800,4294921979,TMC_NAB_D_001,,,11,AUD,2,0,0,1,0,0,NAB18NOV29_2800P.BT8,C,1,NAB18N
OV29_2900P.BV8,B,1,,?,0,,?,0
```

2.4.3 Tick Size Table Entry

This message contains information on a tick size for a price range. Together, all Tick Size messages with the same Order Book ID form a complete Tick Size Table. Each order book has a set of Tick Size Table Entries to define its tick size table.

Name	Length	Value	Notes
Message Type	1	“L”	Tick Size Message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	The order book this entry belongs to (instrument).
Tick Size	8	Numeric	Tick Size for the given price range.
Price From	4	Price	Start of price range for this entry. **Note**: The number of decimal places for prices in ITCH will universally be disseminated with 2 decimal places, regardless of the configured number of decimal places for the instrument in ASX Trade.
Price To	4	Price	End of price range for this entry. **Note**: The number of decimal places for prices in ITCH will universally be disseminated with 2 decimal places, regardless of the configured number of decimal places for the instrument in ASX Trade.

Tick Size Table Entry Example Message

Tick table with 3 ranges.

```
L,911814500,85603,10,10,990
L,911814500,85603,50,1000,19990
L,911814500,85603,100,20000,2147483600
```

2.5 Event and State Change Messages

2.5.1 System Event Message

The system event message type is used to signal a market or data feed handler event. The format is specified below.

Name	Length	Value	Notes
Message Types	1	“S”	System Event Message
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Event Code	1	Alpha	See System Event Codes below.

The system supports the following event codes on a daily basis.

Code	Explanation
“O”	**Start of Messages** Outside of time stamp messages, the start of day message is the first message sent every trading day.

“C”	**End of Messages** This is always the last message sent every trading day.

2.5.2 Order Book State Message

The Order Book State message relays information on session state changes.

Name	Length	Value	Notes
Message Types	1	“O”	Order Book State Message
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	Order Book Identifier (instrument).
State Name	20	Alpha	Name of the session state.

Order Book State Change Example Message

Indicates stock is in the CLOSE state.

```
O,911814500,85603,CLOSE
```

2.6 Market by Order Messages

Note:
Order IDs are only unique per order book and side. When modifying or deleting orders, be careful to only update the order of the applicable side and order book, since the same Order ID may be present in multiple order books and/or sides.

2.6.1 Add Order Messages

An Add Order Message indicates that a new order has been accepted by ASX Trade and was added to the lit order book. The message includes an Order ID that is unique per order book and side.

ASX ITCH supports two variations of the Add Order message.

2.6.1.1 Add Order – No Participant ID

This message will be generated for anonymous instruments in ASX Trade.

Name	Length	Value	Notes
Message Type	1	“A”	Add Order Message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order ID	8	Numeric	The identifier assigned to the new order. Note that the number is *only* unique per order book and side.
Order Book ID	4	Numeric	Order Book Identifier (instrument).

Side	1	Alpha	The type of order being added. Values:
			B = buy order
			S = sell order.

Order Book Position	4	Numeric	Rank within the order book. See Building an Order Book View for details.
Quantity	8	Numeric	The visible quantity of the order.

Name	Length	Value	Notes
			Note: Orders with an undisclosed quantity will have this field set to 0.
Price	4	Price	The price of the new order. Refer to Data Types for field processing notes.
Exchange Order Type	2	Numeric	Additional order attributes. Values: 4 = Market Bid 8 = Price Stabilisation 32 = Undisclosed 8192 = Implied order **Note**: This field is a bit map. Multiple values may be set simultaneously.
Lot Type	1	Numeric	Lot Type. Values: 0 = Undefined 1 = Odd Lot 2 = Round Lot 3 = Block Lot 4 = All or None Lot.

Add Order without Customer ID Example Message

```
A,143460200,621f1282:0000e5ed,85603,B,32,1,100,0,2
```

2.6.1.2 Add Order With Participant ID

This message will be generated for non-anonymous instruments in ASX Trade.

Name	Length	Value	Notes
Message Type	1	"F"	Add Order Message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order ID	8	Numeric	The identifier assigned to the new order. Note that the number is *only* unique per order book and side.
Order Book ID	4	Numeric	Order Book Identifier (instrument).

Side	1	Alpha	The type of order being added. Values:
			B = buy order
			S = sell order.

Order Book Position	4	Numeric	Rank within order book. See *Building an Order Book View* for details.
Quantity	8	Numeric	The visible quantity of the order. **Note**: Orders with an undisclosed quantity will have this field set to 0.
Price	4	Price	The price of the new order. Refer to Data Types for field processing notes.
Exchange Order Type	2	Numeric	Additional order attributes. Values: 4 = Market Bid 8 = Price Stabilisation 32 = Undisclosed

Name	Length	Value	Notes
			8192 = Implied order **Note**: This field is a bit map. Multiple values may be set simultaneously.
Lot Type	1	Numeric	Lot Type. Values: 0 = Undefined 1 = Odd Lot 2 = Round Lot 3 = Block Lot 4 = All or None Lot.
Participant ID	7	Alpha	Participant identifier associated with the entered order.

Add Order with Participant ID Example Message

```
F,434545900,621f1282:00012165,9916914,S,1,2,649900,8192,2,AU550
```

2.6.2 Modify Order Messages

Modify Order messages always include the Order ID, Order Book ID and Side of the order to which the update applies. To determine the visible quantity of an order, ASX ITCH subscribers must deduct the executed quantity stated in the Modify Order message from the original quantity stated in the Add Order message with the same Order ID. ASX ITCH may send multiple Modify Order messages for the same order and the effects are cumulative. When the visible quantity for an order reaches zero, the order should be removed from the order book.

2.6.2.1 **Order Executed Message**

This message is sent whenever an order in the book is executed in whole or in part.

If the incoming order causing the match cannot be fully filled, the remainder will be placed in the order book after the match has occurred.

It is possible to receive several Order Executed Messages for the same order if that order is executed in several parts. Multiple Order Executed Messages on the same order are cumulative.

Name	**Length**	**Value**	**Notes**
Message Type	1	“E”	Order Executed message
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order ID	8	Numeric	The order ID associated with the executed order.
Order Book ID	4	Numeric	Order Book Identifier (instrument).

Side	1	Alpha	The type of order being executed. Values:
			B = buy order
			S = sell order.

Executed Quantity	8	Numeric	The traded quantity.
Match ID	12	Numeric	Assigned by the system to each match executed.
Participant ID, owner	7	Alpha	Participant identifier of the owner of the order. Blank for anonymous instruments.
Participant ID, counterparty	7	Alpha	Participant identifier of the counterparty to the execution. Blank for anonymous instruments.

Order Executed Example Message

```
E,467345200,621f1282:00012504,6640114,S,5,00fb30c2:0000003d:00000001,AU551,AU550
```

2.6.2.2 Order Executed with Price Message

This message is sent when an order in the book is executed in whole or in part with a price different than the initial display price.

If the incoming order causing the match cannot be fully filled, the remainder will be placed in the order book after the match has occurred.

It is possible to receive several Order Executed messages for the same order if that order is executed in several parts. Multiple Order Executed messages on the same order are cumulative.

The executions may be marked as non-printable. If a customer is looking to use the ASX ITCH data in trade tickers or volume calculations, ASX recommends that customers ignore messages marked as non-printable to prevent double counting.

Name	Length	Value	Notes
Message Type	1	"C"	Order Executed message
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order ID	8	Numeric	The order ID associated with the executed order.
Order Book ID	4	Numeric	Order Book Identifier (instrument).

Side	1	Alpha	The type of order being executed. Values:
			B = buy order
			S = sell order.

Executed Quantity	8	Numeric	The traded quantity.
Match ID	12	Numeric	Assigned by the system to each match executed.
Participant ID, owner	7	Alpha	Participant identifier of the owner of the order. Blank for anonymous instruments.
Participant ID, counterparty	7	Alpha	Participant identifier of the counterparty to the execution. Blank for anonymous instruments.
Trade Price	4	Price	The traded price.
Occurred at Cross	1	Alpha	Indicates if an order was executed in an auction. Values: Y = order executed in an auction N = order executed in continuous matching.
Printable	1	Alpha	Indicates if the execution should be included in trade tickers and volume calculations. Values: N = do not include in trade tickers and volume calculations Y = include in trade tickers and volume calculations.

Order Executed at Price Example Message

```
C,301652700,621f1282:000120ac,4294922738,B,10,00fb30c2:00000036:00000001,AU550,AU551,50,N
,N
```

2.6.2.3 Order Replace Message

This message is sent when changes are made to an order where the public order id remains the same, such as undisclosed order trading below the undisclosed threshold, or a change in the undisclosed state of an order.

The Side, Order Book ID, and Participant Identifier (if non-anonymous) remain the same as the original order. Customer identifiers are not part of the Order Replace message.

Name	Length	Value	Notes
Message Type	1	"U"	Order Replace message
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order ID	8	Numeric	The original order identifier of the order being replaced. Note that the Order ID is only unique per order book and side. **Note:** The Order ID does not change when the order is replaced.
Order Book ID	4	Numeric	Order Book identifier (instrument).

Side	1	Alpha	The type of order being replaced. Values:
			B = buy order
			S = sell order.

New Order Book Position	4	Numeric	New rank within the order book. See Building an Order Book View for more information.
Quantity	8	Numeric	The new visible quantity of the order. **Note**: Orders with an undisclosed quantity will have this field set to 0.
Price	4	Price	The new price of the order.
Exchange Order Type	2	Numeric	Additional order attributes. Values: 4 = Market Bid 8 = Price Stabilisation 32 = Undisclosed. **Note**: This field is a bit map. Multiple values may be set simultaneously.

Order Replace Example Message

Undisclosed order added with A. U message makes it undisclosed and changes quantity to 2,000

```
A,427482900,621f1281:00014a14,70669,B,2,0,270300,32,2
U,465706300,621f1281:00014a14,70669,B,1,2000,270300,0
```

2.6.2.4 Order Delete Message

This message is sent whenever an order in the book is deleted. There will be no remaining quantity, so the order should be removed from the book.

Please note that normally no Order Delete message is sent when an order is completely filled. The receiver needs to keep track of the remaining quantity on all orders by recalculating the remaining quantity on each Order Executed message received. Orders must be removed from the book when remaining quantity reaches zero.

Note:
Order Delete messages are sent when orders with undisclosed quantity are fully filled.

Note:
Order Delete messages are sent when orders are inactivated. When central inactive orders are reactivated by a customer, this order will be added as a new order (Add Order message) with the same Order ID.

Name	Length	Value	Notes
Message Type	1	“D”	Order Delete message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order ID	8	Numeric	The ID of the order being deleted. **Note**: The Order ID is only unique per order book and side.
Order Book ID	4	Numeric	Order Book identifier (instrument).

Side	1	Alpha	The type of order being deleted. Values:
			B = buy order
			S = sell order.

Order Delete Example Message

```
D,674262000,621f1282:00012165,9916914,S
```

2.7 Trade Messages

The Trade Message is designed to provide details for executions in dark order books and reporting the individual legs of traded combinations.

Since no Add Order Message is generated when a dark order is initially entered, the Order Executed message cannot be used for those matches. The Trade Message is used to report a match for a non-displayable order in the book.

It is possible to receive multiple Trade Messages for the same order if that order is executed in several parts. Trade Messages for the same order are cumulative.

Trade Messages should be included in trade tickers as well as volume and other market statistics when they have the Printable flag set to Y. Since Trade Messages do not affect the displayed book, they may be ignored by customers just looking to build and track the order book view.

Name	Length	Value	Notes
Message Type	1	“P”	Trade message identifier
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Match ID	12	Numeric	The unique reference number assigned to the order on the book being executed.

Side	1	Alpha	The type of non-display order being matched. Values:
			B =buy order
			S =sell order.
			Side can be set to blank e.g. for Centre Point trades.

Quantity	8	Numeric	The quantity being matched in this execution.
Order Book ID	4	Numeric	Order Book identifier (instrument).
Trade Price	4	Price	The price of the trade.
Participant ID, owner	7	Alpha	Participant identifier of the owner of the order. Blank for anonymous instruments.
Participant ID, counterparty	7	Alpha	Participant identifier of the counterparty to the execution. Blank for anonymous instruments and certain TMC executions.

Name	Length	Value	Notes
Printable	1	Alpha	Indicates if the execution should be included in trade tickers and volume calculations. Values: N = Do not include in trade tickers and volume calculations Y = Include in trade tickers and volume calculations.
Occurred at Cross	1	Alpha	Values: Y = Yes, trade occurred in an auction. N = No, trade occurred in continuous matching.

Trade Example Message

```
P,465706300,00fb30c1:00000009:00000002,,48000,70669,270300,,,Y,N
```

2.8 Auction Messages – Equilibrium Price Update

This message is used when auctions occur. The message provides the changes in equilibrium (auction) price. Note that subtracting the Ask Quantity from the Bid Quantity will yield the Surplus Volume.

Name	Length	Value	Notes
Message Type	1	"Z"	Equilibrium Price Update message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	Order Book identifier (instrument).
Bid Quantity	8	Numeric	Total Bid Quantity available for execution.
Ask Quantity	8	Numeric	Total Ask Quantity available for execution.
Equilibrium Price	4	Price	The price at which matching will occur.
Best Bid Price	4	Price	Best Bid Price.
Best Ask Price	4	Price	Best Ask Price.
Best Bid Quantity	8	Numeric	Quantity at Best Bid Price.
Best Ask Quantity	8	Numeric	Quantity at Best Ask Price.

Equilibrium Price Update Example Messages

Z message pre auction

```
Z,408258000,70639,66,53,26100,26100,26100,66,53
```

Z message on the open

```
Z,030441800,70639,0,0,-2147483648,-2147483648,-2147483648,0,0
O,030441800,70639,OPEN
```

2.9 Building an Order Book View

The information needed to build an order book view from the ASX ITCH message flow is contained in Add Order Messages and Modify Order Messages. The messages are:

- Add Order – No Participant ID
- Add Order – With Participant ID
- Order Executed
- Order Executed with Price

- Order Replace
- Order Delete.

The two variants of the Add Order messages have the same meaning; an order is added to the book. Orders are ranked by:

- Price
- Order Book Position - “1” denotes the highest ranked order. For an Order Replace, the order must be removed from its previous position and inserted at the new Order Book Position. An order inserted at an existing position shifts the order in that position down (and all orders below as well). A deleted or fully filled order causes existing orders below it to shift their position up one step to fill the “void”.

The Order Executed and Order Executed with Price messages signal a partial or full fill. The order quantity must be reduced by the quantity of the message.

The Order Replace message signals that the order has been modified. The current rank may or may not be lost in the process. The Order Book Position will show the new rank within the book.

The Order Delete message tells the recipient to remove the referenced order.

2.10 Building a Trade Ticker

The Trade Ticker is based on the following messages:

- Order Executed
- Order Executed with Price
- Trade.

Note:
Trades and Order Executed with Price messages marked as non-printable should be excluded to avoid double booking of trades.

Note:
Trade Reports are not included in ASX ITCH.

3 Glimpse

3.1 Architecture

The Glimpse data feed is made up of a series of sequenced messages. Each message is variable in length based on the message type. The messages will be binary encoded using SoupBinTCP. SoupBinTCP takes care of sequencing and delivery guarantees.

Note:
Customers must log in with SoupBinTCP sequence number 1 to correctly receive data.

3.1.1 Protocol

Name	Length
SoupBinTCP	SoupBinTCP is a lightweight point-to-point protocol, built on top of TCP/IP sockets that allow delivery of a set of sequenced messages from a server to a customer in real-time. SoupBinTCP guarantees that the customer receives each message generated by the server in sequence, even across underlying TCP/IP socket connection failures. The sequence numbers are implicit, meaning that the customer maintains a counter that is increased every time a message is received. To reconnect after a connection loss, the customer submits the last seen sequence number in its Logon message, and the server resends every message starting from that sequence number. **Note**: Refer to the Nasdaq Trader website (https://www.nasdaqtrader.com) for the latest SoupBinTCP manual.

3.2 Data Types

Glimpse messages have the same data types as ASX ITCH messages.

3.3 Glimpse Messages

Glimpse uses a subset of ASX ITCH messages.

Glimpse utilises all messages used by ASX ITCH to describe the current picture of the book, with the exception of those messages that change the picture of the order book (e.g. Delete, Execute and Trade).

3.3.1 End of Snapshot Message

The end of snapshot message is sent at the end of a Glimpse snapshot and returns the current ASX ITCH sequence number to be used when connecting to the ASX ITCH feed.

To maintain a real-time order display, real-time processing of ASX ITCH messages should begin with the sequence number stated in this snapshot message.

Name	Length	Value	Notes
Message Type	1	“G”	End of Snapshot Message.
Sequence Number	20	Alpha	ASX ITCH sequence number when the snapshot was taken. To be used when applying messages received via the ASX ITCH MoldUDP feed. **Note**: While Glimpse is a binary feed, the SoupBinTCP uses ASCII characters to represent the sequence number.

4 Appendix 1 – ASX ITCH Scenarios

To illustrate ASX ITCH functionality a number of scenarios are provided.

4.1 Normal Orders

4.1.1 New order entered, amended and deleted

A new order is entered, amended and deleted without trading

Step	Comments and examples
Order entered into book	• A (or F) message to add the order Buy of 100 at $5.50 `A,251202700,62018a81:000119dd,70578,B,1,100,55000,0,2`

Price and volume of order amended	• D message to remove order
	• A (or F) message to add amended order details.
	
	Amended to 90 at $5.51
	
	`D,029312500,62018a81:000119dd,70578,B`
	`A,029312500,62018a81:000119dd,70578,B,1,90,55100,0,2`

Order deleted	`D,128252200,62018a81:000119dd,70578,B`

4.1.2 New order entered, fully trades

A new order is entered and fully trades on entry

Step	Comments and examples

Order entered	• E message for the amend of existing order(s) that the incoming order trades against
	
	Buy of 200 at $6.28 enters and trades against 3 existing orders of 50, 125 and 25
	
	`E,011828600,6202f281:00014039,72007,S,50,00fae8c1:000000b0:00000001,,`
	`E,011828600,6202f281:00014072,72007,S,125,00fae8c1:000000b0:00000002,,`
	`E,011828600,6202f281:00014091,72007,S,25,00fae8c1:000000b0:00000003,,`

4.1.3 New order entered, partially trades

A new order is entered and partially trades on entry with existing orders

Step	Comments and examples

Order entered	• Modification to the existing orders are published using E messages
	• The residual amount of the incoming order is communicated via an A message
	
	Sell of 154 at $0.235 enters and trades against 2 existing orders of 75 and 52
	
	`E,078664300,62045a81:000128dc,70646,B,75,00faec41:000000e1:00000001,,`
	`E,078664300,62045a81:000128e1,70646,B,52,00faec41:000000e1:00000002,,`
	`A,078664300,62045a81:000128e4,70646,S,1,27,2350,0,2`

4.1.4 Existing order amended, and fully trades

An existing order is amended and fully trades as a result of the amendment

Step	Comments and examples

Order amended	• Existing buy order is removed from the market using a D message
	• E messages communicate the changes to the sell orders that trade with the buy.
	
	Buy of 115 amended to $1.80. Trades against two orders of 50 and 65.
	`D,701476200,62045a81:000270d6,72112,B`
	`E,701476200,62045a81:00027fbe,72112,S,50,00faec41:00000108:00000001,,`
	`E,701476200,62045a81:00028a2d,72112,S,65,00faec41:00000108:00000002,,`

4.1.5 Existing order amended, and partially trades

An existing order is amended and partially trades as a result of the amendment

Step	Comments and examples

Order amended	• Existing buy order is removed from the market using a D message.
	• E messages communicate the changes to the sell orders that trade with the buy.
	• Residual 114 of buy order is added into the book using an A message.
	
	Buy of 170 amended to $3.37. Trades against two orders of 45 and 11.
	`D,533484100,62089281:00017bd0,70579,B`
	`E,533484100,62089281:00017be5,70579,S,11,00faf701:000000cc:00000001,,`
	`E,533484100,62089281:00017c24,70579,S,45,00faf701:000000cc:00000002,,`
	`A,533484100,62089281:00017bd0,70579,B,1,114,33700,0,2`

4.1.6 Orders trade in auction

Orders are entered in pre-open and trade in the auction

Step	Comments and examples

Orders entered at best bid	• Z message for equilibrium price update
	• A message to add order
	
	Buy orders of 34 and 32 at $2.61 entered
	`Z,618782100,70639,0,0,-2147483648,26100,-2147483648,34,0`
	`A,618782100,621f1281:00010901,70639,B,1,34,26100,0,2`
	`Z,080693700,70639,0,0,-2147483648,26100,-2147483648,66,0`
	`A,080693700,621f1281:00010902,70639,B,2,32,26100,0,2`

Orders entered below best bid	• A message to add order
	
	Buy orders or 32 and 35 at $2.59 entered
	`A,608788700,621f1281:00010903,70639,B,3,32,25900,0,2`
	`A,351494800,621f1281:00010904,70639,B,4,35,25900,0,2`

Orders entered at best ask	• Z message for equilibrium price update • A message to add order

Step	Comments and examples

	Sell orders of 53 and 75 entered at $2.61
	`Z,408258000,70639,66,53,26100,26100,26100,66,53`
	`A,408258000,621f1281:00010905,70639,S,1,53,26100,0,2`
	`Z,169307200,70639,66,128,26100,26100,26100,66,128`
	`A,169307200,621f1281:00010906,70639,S,2,75,26100,0,2`

Stock moves to open	• Z message indicating equilibrium no longer exists as auction has occurred
	• C message communicating alterations to orders that traded in auction
	• O message indicating the stock has moved to open
	
	From the orders shown above, the first two buy orders trade fully, against the first and part of the second sell orders, at the equilibrium price of $2.61
	`Z,030441800,70639,0,0,-2147483648,-2147483648,-2147483648,0,0`
	`C,030441800,621f1281:00010901,70639,B,34,00fb30c1:00000003:00000001,,,26100,Y,N`
	`C,030441800,621f1281:00010905,70639,S,34,00fb30c1:00000003:00000001,,,26100,Y,Y`
	`C,030441800,621f1281:00010905,70639,S,19,00fb30c1:00000003:00000002,,,26100,Y,Y`
	`C,030441800,621f1281:00010902,70639,B,19,00fb30c1:00000003:00000002,,,26100,Y,N`
	`C,030441800,621f1281:00010902,70639,B,13,00fb30c1:00000003:00000003,,,26100,Y,N`
	`C,030441800,621f1281:00010906,70639,S,13,00fb30c1:00000003:00000003,,,26100,Y,Y`
	`O,030441800,70639,OPEN`

4.2 Undisclosed Order, Continuous Trading

4.2.1 Undisclosed Order Entered Into Book

An undisclosed order is entered and rests in the book.

Step	Comments and examples
Order entered	• A message to add order Undisclosed buy order of 500,000 entered at $4.82 `A,316294200,620b6281:00015afd,70654,B,1,0,48200,32,2`

4.2.2 New Order Fully Trades with Existing Undisclosed Order – 1

New order entered and trades fully with an existing undisclosed order. The undisclosed order remains above the undisclosed threshold.

Step	Comments and examples
Order entered	• P message communicates the trade Sell order of 75 at $4.82 trades against existing undisclosed of $4.82 `P,818867900,00fafe41:000003f3:00000001,,480000,70654,48200,,,Y,N`

4.2.3 New Order Fully Trades with Existing Undisclosed Order – 2

New order entered and trades fully with an existing undisclosed order. The undisclosed order falls below the undisclosed threshold.

Step	Comments and examples

Order entered	• P message communicates the trade
	• U message communicates undisclosed order volume being disclosed
	
	Sell order of 480,000 at $4.82 trades against existing undisclosed of $4.82. Residual volume is 19925
	
	`P,818867900,00fafe41:000003f3:00000001,,480000,70654,48200,,,Y,N`
	`U,818867900,620b6281:00015afd,70654,B,1,19925,48200,0`

4.2.4 New Order Fully Trades with Existing Undisclosed Order – 3

New order entered and trades fully with an existing undisclosed order. The undisclosed order fully trades.

Step	Comments and examples

Order entered	• P message communicates the trade
	• D message communicates removal of undisclosed order
	
	Sell order of 250,000 at $4.83 trades against existing undisclosed of 250,000 at $4.83.
	
	`P,652759800,00fafe41:00000409:00000001,,250000,70654,48300,,,Y,N`
	`D,652759800,620b6281:00015d33,70654,B`

4.3 Iceberg Order Execution, Continuous Trading

4.3.1 New Order Fully Trades with Existing Iceberg Order - 1

New order, with a quantity less than the shown quantity of an existing iceberg order, fully trades with an existing iceberg order.

Step	Comments and examples
Order entered	• E message communicates the alteration to the resting iceberg order Sell order of 350 trades against an iceberg order with a shown quantity of 1,000 `E,366535700,620b6281:00017b2c,70717,B,650,00fafe41:00000545:00000001,,`

4.3.2 New Order Fully Trades with Existing Iceberg Order – 2

New order, with a quantity equal to the shown quantity of an existing iceberg order, fully trades with an existing iceberg order.

Step	Comments and examples
Order entered	• P message communicates the trade for the shown quantity. Sell order of 1,000 trades against an iceberg order with a shown quantity of 1,000 `P,698517600,00fafe41:00000549:00000001,,1000,70717,31900,,,Y,N`

4.3.3 New Order Fully Trades with Existing Iceberg Order – 3

New order, with a quantity greater than the shown quantity of an existing iceberg order, but less than the total quantity, fully trades with an existing iceberg order and other orders in the book.

Step	Comments and examples

Order entered	• D message removes iceberg order
	• P message for the trade of the incoming order against the iceberg order
	• E messages communicate alterations to the existing, non-iceberg orders
	• A message adds the next refresh of the iceberg order
	
	Sell order of 650 trades against an iceberg order with a current shown quantity of 350, refresh quantity of 1,000. 3 non-iceberg orders of 100, 125 and 75 fully fill the sell order
	`D,323343400,620b6281:00017b2c,70717,B`
	`P,323343400,00fafe41:00000548:00000001,,350,70717,31900,,,Y,N`
	`E,323343400,620b6281:00017b66,70717,B,100,00fafe41:00000548:00000002,,`
	`E,323343400,620b6281:00017b69,70717,B,125,00fafe41:00000548:00000003,,`
	`E,323343400,620b6281:00017b6a,70717,B,75,00fafe41:00000548:00000004,,`
	`A,323343400,620b6281:00017b2c,70717,B,1,1000,31900,0,2`

4.3.4 New Order Fully Trades with Existing Iceberg Order - 4

New order fully trades out an existing iceberg order i.e. it has a quantity of the total remaining iceberg order.

Step	Comments and examples

Order entered	• P message for the trade of the incoming order against the iceberg order
	• D message removes iceberg order
	
	Sell order of 8,000 trades against an iceberg order with a total quantity of 8,000, refresh quantity of 1,000.
	`P,581539800,00fafe41:0000054a:00000001,,8000,70717,31900,,,Y,N`
	`D,581539800,620b6281:00017b2c,70717,B`

4.4 Tailor Made Combinations

4.4.1 New TMC Order Trades with Existing TMC Order

TMC order entered and trades with another TMC order

Step	Comments and examples

TMC created	• M message describes TMC
	• L message communicates tick table for TMC
	• O message communicates state of the TMC
	
	TMC created to buy MQG18NOV29_8000C.Z78 and sell MQG18NOV29_8100C.Z98
	`M,947193200,4294922738,TMC_MQG_D_001,,,11,AUD,2,0,0,1,0,0,MQG18NOV29_8000C.Z78,B,1,MQG18NOV29_8100C.Z98,C,1,,?,0,,?,0`
	`L,947193200,4294922738,50,-2147483600,2147483600`
	`O,947193200,4294922738,OPEN`

Buy TMC order entered	• A (or F) message to add order

Step	Comments and examples
	Buy order of 10 at $0.005 entered `F,485087500,621f1282:000120ac,4294922738,B,1,10,50,0,2,AU550`

Sell TMC entered and matches against buy order	• C message communicating alterations to the resting order
	• P message for trades in underlying stocks
	
	Order above fully trades with incoming order. The incoming order also fully trades
	
	`C,301652700,621f1282:000120ac,4294922738,B,10,00fb30c2:00000036:000000`
	`01,AU550,AU551,50,N,N`
	`P,301652700,00fb30c2:00000036:00000001,S,10,10310130,10050,AU551,AU550`
	`,Y,N`
	`P,301652700,00fb30c2:00000036:00000001,S,10,10441202,10000,AU550,AU551`
	`,Y,N`

4.4.2 New TMC Order Trades with Existing Orders in the Book

TMC order rests in the book and trades with orders in underlying books.

Step	Comments and examples

TMC created	• M message describes TMC
	• L message communicates tick table for TMC
	• O message communicates state of the TMC
	
	TMC created to buy MQG18NOV29_8600C.YK8 and sell MQG18NOV29_8700C.Z18
	
	`M,347145500,4294857202,TMC_MQG_D_002,,,11,AUD,2,0,0,1,0,0,MQG18NOV29_8`
	`600C.YK8,B,1,MQG18NOV29_8700C.Z18,C,1,,?,0,,?,0`
	`L,347145500,4294857202,50,-2147483600,2147483600`
	`O,347145500,4294857202,OPEN`

Buy TMC order entered	• A (or F) message to add order Buy order of 15 at $0.01 entered `F,745001900,621f1282:00012165,4294857202,B,1,15,100,0,2,AU550`

Orders placed in one of the underlying books	• A (or F) to add the order in the underlying book
	• A (or F) to add the implied orders in other underlying book
	• D message to remove implied orders in other underlying book
	
	3 orders added of 2, 5 and 3 at 6500
	
	`F,434545900,621f1282:00012165,9916914,S,1,2,649900,8192,2,AU550`
	`F,434545900,621f1282:00012503,6640114,S,1,2,650000,0,2,AU551`
	
	`D,315523800,621f1282:00012165,9916914,S`
	`F,315523800,621f1282:00012165,9916914,S,1,7,649900,8192,2,AU550`
	`F,315523800,621f1282:00012504,6640114,S,2,5,650000,0,2,AU551`
	
	`D,674262000,621f1282:00012165,9916914,S`
	`F,674262000,621f1282:00012165,9916914,S,1,10,649900,8192,2,AU550`
	`F,674262000,621f1282:00012506,6640114,S,3,3,650000,0,2,AU551`

Step	Comments and examples

Incoming order in underlying book triggers match	• E message communicates change in orders in underlying book(s)
	• E message communicates change in implied order(s)
	• C communicates execution of TMC order
	
	An incoming order in the underlying book with implied orders matches against the implied order, causing the TMC to fully trade
	`E,467345200,621f1282:00012503,6640114,S,2,00fb30c2:0000003d:00000001,AU551,AU550`
	`E,467345200,621f1282:00012504,6640114,S,5,00fb30c2:0000003d:00000001,AU551,AU550`
	`E,467345200,621f1282:00012506,6640114,S,3,00fb30c2:0000003d:00000001,AU551,AU550`
	`C,467345200,621f1282:00012165,4294857202,B,10,00fb30c2:0000003d:00000001,AU550,,100,N,N`
	`E,467345200,621f1282:00012165,9916914,S,10,00fb30c2:0000003d:00000001,AU550,AU551`

4.4.3 TMC Order Trades with an Opposing TMC Order during an Auction

TMC orders are entered in pre-open and trade in the auction

Step	Comments and examples

TMC created	• M message describes TMC
	• L message communicates tick table for TMC
	• O message communicates state of the TMC
	
	TMC created to buy CBA and sell WBC
	`M,172637200,4294921827,TMC_CBA_E_001,,,11,AUD,2,0,0,1,0,0,CBA,B,1,WBC,C,1,,?,0,,?,0`
	`L,172637200,4294921827,100,-2147483600,2147483600`
	`O,172637200,4294921827,PRE_OPEN`

Order entered at best bid	• Z message for equilibrium price update
	• A (or F) message to add order
	
	Buy order of 10 $0.02 entered
	`Z,648537000,4294921827,0,0,-2147483648,200,-2147483648,10,0`
	`A,648537000,621f1282:0000ff29,4294921827,B,1,10,200,0,2`

Order entered at best ask	• Z message for equilibrium price update
	• A (or F) message to add order
	
	Sell order of 10 at $0.02 entered
	`Z,040753000,4294921827,10,10,200,200,200,10,10`
	`A,040753000,621f1282:0000ff2b,4294921827,S,1,10,200,0,2`

Stock moves to open	• Z message indicating equilibrium no longer exists as auction has occurred • C message communicating alterations to orders that traded in auction • P message for trades in underlying stocks • O message indicating the stock has moved to open

Step	Comments and examples

	Both orders entered above fully trade
	`Z,030598400,4294921827,0,0,-2147483648,-2147483648,-2147483648,0,0`
	`C,030598400,621f1282:0000ff29,4294921827,B,10,00fb30c2:00000000:00000001,,,200,Y,N`
	`C,030598400,621f1282:0000ff2b,4294921827,S,10,00fb30c2:00000000:00000001,,,200,Y,N`
	`P,030598400,00fb30c2:00000000:00000001,,10,85603,578300,,,Y,N`
	`P,030598400,00fb30c2:00000000:00000001,,10,85837,578100,,,Y,N`
	`O,030598400,4294921827,OPEN`

4.5 Miscellaneous

4.5.1 Standard Instrument Creation

An equity product is created at system start up

Step	Comments and examples

Instrument created	• R message describes the instrument
	• L message communicates tick table
	• O message communicates state
	
	Instrument CBA created
	`R,911814500,85603,CBA,CWLTH BANK FPO [CBA],AU000000CBA7,5,AUD,2,0,0,1,0,0`
	`O,911814500,85603,CLOSE`
	`L,911814500,85603,10,10,990`
	`L,911814500,85603,50,1000,19990`
	`L,911814500,85603,100,20000,2147483600`

5 Appendix 2 – ASX MoldUDP64 Protocol Specification

5.1 Overview

MoldUDP64 is a networking protocol that allows efficient and scalable transmission of data messages in a "one transmitter to many listeners" scenario. MoldUDP64 is a lightweight protocol layer built on top of UDP that provides a mechanism for listeners to detect and re-request missed packets.

In MoldUDP64, each outbound packet is transmitted only once regardless of the number of listeners. Multiple messages may also be aggregated into a single network packet to reduce network traffic. Optional caching request servers can be placed nearby remote receivers to reduce latency and bandwidth over WAN links.

This section describes the messages sent between a MoldUDP64 server and its clients. MoldUDP64 transmitters send downstream packets via UDP multicast to transport the normal data stream sent to the listeners. These packets are also sent via UDP unicast in response to a Request message submitted by a listener. MoldUDP64 clients can send these Request messages to request the retransmission of any desired packets from the data stream.

The MoldUDP64 server will transmit on a well-known multicast group for each type of downstream MoldUDP64 data stream on a network. The listeners must subscribe to this multicast group to receive the downstream data. One or more request servers may also be deployed to service any unicast client requests for retransmission of specific messages. The listeners must be configured with these IP addresses and port combinations to which they can submit the requests.

5.2 Assumptions

All number fields in the MoldUDP64 messages specified in this document (i.e. sequence number, message counts and message lengths etc.) are binary numbers formatted in Big Endian mode
(i.e. most significant byte first).

Note:
This need not apply to the data contained the Message Data fields of the Message Blocks.

5.3 Terms

5.3.1 Message

A message is an atomic piece of information carried by the MoldUDP64 protocol.

MoldUDP64 can theoretically handle individual messages from zero bytes up to 64KB in length although individual messages should be kept small enough so that the UDP underlying network protocol can efficiently carry the resulting MoldUDP64 packets.

The contents of a MoldUDP64 message are defined by the higher level application.

5.3.2 Session

A session is a sequence of one or more messages.

While a single session can last indefinitely, typically the application will define a session to logically group messages together based on time delimitation.

Once a session is terminated, no more messages can be sent on that session. Depending on the design of the MoldUDP64 system and the application, receivers may still be able to re- request messages from a terminated session.

A session is considered active if it has started but not yet been terminated.

5.4 Downstream Packet

A MoldUDP64 transmitter sends “downstream” packets that are received by MoldUDP64 listeners. A MoldUDP64 packet may contain a payload of 0 or more data stream messages.

Each MoldUDP64 packet consists of a Downstream Packet Header and of a series of Message Blocks. The Message Blocks carry the actual data of the stream.

5.4.1 Downstream Packet Header

Field Name	Offset	Length	Value	Notes
Session	0	10	ANUM	Indicates the session to which this packet belongs.
Sequence Number	10	8	NUM	The sequence number of the first message in the packet.
Message Count	18	2	NUM	The count of messages contained in this packet.

5.4.2 Sequence Number

The Sequence Number field of the Packet Header indicates the sequence number of the first message in the packet. If there is more than one message contained in a packet, any messages following the first message are implicitly numbered sequentially.

5.4.3 Message Count

The number of Message Blocks contained in a MoldUDP64 packet is specified by the Message Count field of the Packet Header. The maximum payload size of a Downstream Packet is determined by the sender.

Note:
A Message Count of zero denotes a heartbeat and a Message Count of OxFFFF (hex, or 65535 in decimal) denotes end of session

5.4.4 Downstream Packet Message Block

The first field of a Message Block is the two byte Message Length. The remainder of the Message Block is the variable length Message Data field. The first Message Block field will always start immediately following the Header which is 20 bytes from the beginning of the packet. Subsequent Message Blocks will begin after the last byte of the previous Message Block.

Field Name	Offset	Length	Value	Notes
Message Length	*	2	NUM	Indicates the length in bytes of the message contained in the message block.
Message Data	*	*	ANUM	The message data.

* = Variable Values

5.4.5 Message Length

The Message Length is an unsigned binary count representing the number of message data bytes following this Message Length field. The message length field value does not include the two bytes occupied by the message length field. The total size of the message block is the value of the message length field plus two.

5.4.6 Message Data

The Message Data is actual data of the message being transmitted by MoldUDP64. It is variable length and can be zero length. The meaning of the data is application specific.

5.4.7 Heartbeats

Heartbeats are sent periodically by the server so receivers can sense packet loss even during times of low traffic. Typically, these packets are transmitted every 50 milliseconds and contain the next expected Sequence Number. A heartbeat packet is a MoldUDP64 packet with a Message Count of zero.

5.4.8 End of Session

When the current session is complete, Downstream Packets are sent with a Message Count of 0xFFFF (hex, or 65535 in decimal) for a short time in place of heartbeats. These Downstream Packets contain the next expected Sequence Number, just like heartbeats. While the End of Session messages persist, re-requests may be made on the current session. This is the last chance to ensure that all messages have been received.

5.4.9 Request Packet

The Request Packet is sent to request the retransmission of a particular message or group of messages. The request packet is sent to a request server. A receiver may need to send this request when it detects a sequence number gap in received messages. The response to a valid Request Packet is a standard Downstream Packet unicast back to the source of the retransmission request. This allows downstream MoldUDP64 customers to read the retransmitted Downstream Packet in their multicast processing socket if the request was made from that socket. In other words, the client need only have one socket open to listen to the multicast and to process retransmissions, even though the retransmissions are not multicast.

Field Name	Offset	Length	Value	Notes
Session	0	10	ANUM	Indicates the session to which this packet belongs.
Sequence Number	10	8	NUM	First requested sequence number.
Requested Message Count	18	2	NUM	The number of messages requested for retransmission.

5.4.10 Sequence Number

The Sequence Number field of the Packet Header indicates the sequence number of the first message requested.

5.4.11 Requested Message Count

The Message Count indicates how many messages should be retransmitted. If the total size of the requested messages exceeds the maximum payload size of the of one UDP packet, only the number of messages that completely fit will be returned. Additional retransmission requests must be made for the subsequent messages if they are still desired.

5.5 Receiver Example

A typical MoldUDP64 receiver client would be configured with the following parameters:

- The UDP port to listen on and the Multicast group to join
- A list of one or more Request Servers that are available to answer retransmission requests for this stream. Each server is specified as a host IP address and a UDP port to which to send requests.
- A session and sequence number of the next expected message if the client is being restarted.

A typical MoldUDP64 receiver client may have the following process:

1. Subscribe to multicast.
2. Buffer messages received.
3. Connect to Glimpse.
4. Process Glimpse messages.
5. Receive G message with sequence number.

6. Discard ASX ITCH message below sequence number.
7. Process buffered ASX ITCH messages remaining.
8. Continue to process new incoming ASX ITCH messages.
9. Send retransmit request for any missing sequences.

6 Appendix 3 - SoupBinTCP Protocol

6.1 Overview

SoupBinTCP is a lightweight point-to-point protocol, built on top of TCP/IP sockets that allow delivery of a set of sequenced messages from a server to a client in real-time. SoupBinTCP guarantees that the client receives each message generated by the server in sequence, even across underlying TCP/IP socket connection failures.

SoupBinTCP clients can send messages to the server. These messages are not sequenced and may be lost in the case of a TCP/IP socket failure.

SoupBinTCP is ideal for systems where a server needs to deliver a logical stream of sequenced messages to a client in real-time but does not require the same level of guarantees for client generated messages either because the data stream is unidirectional or because the server application generates higher-level sequenced acknowledgments for any important client-generated messages.

SoupBinTCP is designed to be used in conjunction with higher lever protocols that specify the contents of the messages that SoupBinTCP messages deliver. The SoupBinTCP protocol layer is opaque to the higher-level messages.

Note:
Unlike the ASCII version, messages may include any possibly byte.

SoupBinTCP also includes a simple scheme that allows the server to authenticate the client on login.

6.1.1 SoupBinTCP Logical Packets

The SoupBinTCP client and server communicate by exchanging a series of logical packets.

Each SoupBinTCP logical packet has a:

- Two byte big-endian length that indicates the length of rest of the packet (meaning the length of the payload plus the length of the packet type – which is 1)
- Single byte header which indicates the packet type
- Variable length payload.

SoupBinTCP Logical Packet Structure

Two Byte Packet Length	Packet Type	Variable Length Payload

Note:
The SoupBinTCP logical packets do not necessarily map directly to physical packets on the underlying network socket; they may be broken apart or aggregated by the TCP/IP stack.
The SoupBinTCP protocol does not define a maximum payload length.
Unlike the ASCII version, the payload may contain the line feed character or any character.

6.1.2 Protocol Flow

A SoupBinTCP connection begins with the client opening a TCP/IP socket to the server and sending a Login Request Packet. If the login request is valid, the server responds with a Login Accepted Packet and begins sending Sequenced Data Packets. The connection continues until the TCP/IP socket is broken.

Each Sequenced Data Packet carries a single higher-level protocol message. Sequenced Data Packets do not contain an explicit sequence number; instead both client and server compute the sequence number locally by counting messages as they go.

The sequence number of the first sequenced message in each session is always 1.

Typically, when initially logging into a server the client will set the Requested Sequence Number field to 1 and leave the Requested Session field blank in the Login Request Packet. The client will then inspect the Login Accepted Packet to determine the currently active session. Starting at 1, the client begins incrementing its local sequence number each time a Sequenced Data Packet is received. If the TCP/IP connection is ever broken, the client can then re-log into the server indicating the current session and its next expected sequence number. By doing this, the client is guaranteed to always receive every sequenced message in order, despite TCP/IP connection failures.

SoupBinTCP also permits the client to send messages to the server using Un-sequenced Data Packets at any time after the Login Accepted Packet is received. These messages may be lost during TCP/IP socket connection failures.

6.1.3 Heartbeats

SoupBinTCP uses logical heartbeat packets to quickly detect link failures. The server must send a Server Heartbeat packet anytime more than 1 second has passed since the server last sent any data. This ensures that the client will receive data on a regular basis. If the client does not receive anything (neither data nor heartbeats) for an extended period of time, it can assume that the link is down and attempt to reconnect using a new TCP/IP socket.

Similarly, once logged in, the client must send a Client Heartbeat packet anytime more than 1 second has passed since the client last sent anything. If the server doesn't receive anything from the client for an extended period of time (typically 15 seconds), it can close the existing socket and listen for a new connection.

6.1.4 End of Session Marker

The server indicates that the current session has terminated by sending an End of Session Message. This indicates that there will be no more messages contained in this session.

The client will have to reconnect and login with the new Session ID or a blank (space filled) Session ID to begin receiving messages for the next available session.

6.1.5 Data Types

Character data fields are standard ASCII bytes. Integer fields are binary big-endian values.

6.2 SoupBinTCP Packet Types

6.2.1 Debug Packet

A debug packet can be sent by either side of a SoupBinTCP connection at any time. Debug packets are intended to provide human readable text that may aid in debugging problems. Debug Packets should be ignored by both client and server application software.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"+"	Debug Packet.
Text	3	*	Alpha Numeric	Free form human readable text.

* = Variable Values

6.2.2 Logical Packets Sent by a SoupBinTCP Server

6.2.2.1 **Login Accepted Packet**

The SoupBinTCP server sends a Login Accepted Packet in response to receiving a valid Login Request from the client. This packet will always be the first non-debug packet sent by the server after a successful login request.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"A"	Login Accepted Packet.
Session	3	10	Alpha Numeric	The session ID of the session that is now logged into. Left padded with spaces.
Sequence Number	13	20	Numeric	The sequence number in ASCII of the next Sequenced Message to be sent. Left padded with spaces.

6.2.2.2 **Login Rejected Packet**

The SoupBinTCP server sends this packet in response to an invalid Login Request Packet from the client. The server closes the socket connection after sending the Login Reject Packet. The Login Rejected Packet will be the only non-debug packet sent by the server in the case of an unsuccessful login attempt.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"J"	Login Rejected Packet.
Reject Reason Code	3	10	Alpha Numeric	See Login Reject Codes below.

Login Reject Codes

Code	Explanation
"A"	Not Authorised. There was an invalid username and password combination in the Login Request Message.
"S"	Session not available. The Requested Session in the Login Request Packet was either invalid or not available.

6.2.2.3 **Sequenced Data Packet**

The Sequenced Data Packets act as an envelope to carry the actual sequenced data messages that are transferred from the server to the client. Each Sequenced Data Packet carries one message from the higher-lever protocol. The sequence number of each message is implied; the initial sequence number of the first Sequenced Data Packet for a given TCP/IP connection is specified in the Login Accepted Packet and the sequence number increments by 1 for each Sequenced Data Packet transmitted.

Since SoupBinTCP logical packets are carried via TCP/IP sockets, the only way logical packets can be lost is in the event of a TCP/IP socket connection failure. In this case, the client can reconnect to the server and request the next expect sequence number and pick up where it left off.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"S"	Sequenced Data Packet.
Message	3	*	Anv	Defined by a higher-level protocol, but must not contain any embedded linefeeds.

* = Variable values

6.2.2.4 **Server Heartbeat Packet**

The server should send a Server Heartbeat Packet anytime more than 1 second passes where no data has been sent to the client. The client can then assume that the link is lost if it does not receive anything for an extended period of time.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"H"	Server Heartbeat Packet.

6.2.2.5 **End of Session Packet**

The server will send an End of Session Packet to denote that the current session is finished. The connection will be closed shortly after this packet, and the customer will no longer be able to reconnect to the current session.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"Z"	End of Session Packet

6.2.3 Logical Packets Sent by the SoupBinTCP Client

6.2.3.1 **Login Request Packet**

The SoupBinTCP client must send a Login Request Packet immediately upon establishing a new TCP/IP socket connection to the server.

Client and server must have mutually agreed upon the username and password fields. They provide simple authentication to prevent a client from inadvertently connecting to the wrong server.

Both username and password are case-insensitive and should be padded on the right with spaces.

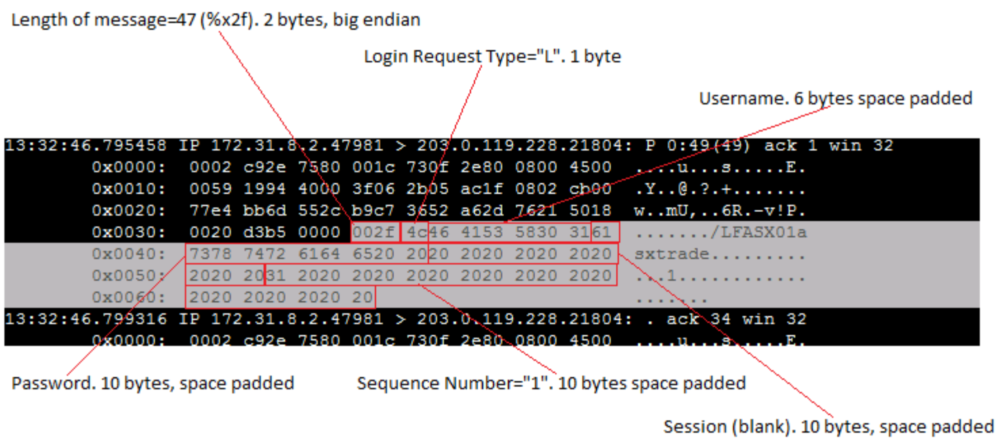
The server can terminate an incoming TCP/IP socket if it does not receive a Login Request Packet within a reasonable period of time (typically 30 seconds).

Login Request Packet

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"L"	Login Request Packet
Username	3	6	Alpha-numeric	Username
Password	9	10	Alpha-numeric	Password
Requested Session	19	10	Alpha-numeric	Specifies the session the client would like to log into, or all blanks to log into the currently active session.
Requested Sequence Number	29	20	Numeric	Specifies the next sequence number in ASCII the client wants to receive upon connection, or 0 to start receiving the most recently generated message.

Glimpse Login

The following is a tcpdump capture displaying the sequence of bytes sent for a successful Glimpse Login.


Length of message=47 (%x2f). 2 bytes, big endian
Login Request Type="L". 1 byte
Username. 6 bytes space padded
Password. 10 bytes, space padded
Sequence Number="1". 10 bytes space padded
Session (blank). 10 bytes, space padded

```
13:32:46.795458 IP 172.31.8.2.47981 > 203.0.119.228.21804: P 0:49(49) ack 1 win 32
        0x0000:  0002 c92e 7580 001c 730f 2e80 0800 4500  ....u...s.....E.
        0x0010:  0059 1994 4000 3f06 2b05 ac1f 0802 cb00  .Y..@.?.+.......
        0x0020:  77e4 bb6d 552c b9c7 3652 a62d 7621 5018  w..mU,..6R.-v!P.
        0x0030:  0020 d3b5 0000 002f 4c46 4153 5830 3161  ......./LFASX01a
        0x0040:  7378 7472 6164 6520 2020 2020 2020 2020  sxtrade.........
        0x0050:  2020 2031 2020 2020 2020 2020 2020 2020  ...1............
        0x0060:  2020 2020 2020 20                        .......
13:32:46.799316 IP 172.31.8.2.47981 > 203.0.119.228.21804: . ack 34 win 32
        0x0000:  0002 c92e 7580 001c 730f 2e80 0800 4500  ....u...s.....E.
```

Note:
The spaces required after the sequence number to fill the packet, and ensure numbers are left justified.

ASX ITCH uses big endians.

Big endian stores the most significant to the least significant bytes from left (lowest address) to right, i.e. it stores 47 (%x2f) in a short (16 bit) number as 00-2f.

This differs to a Little-endian which stores the least significant in the left, up to the most significant byte in the right, i.e. 2f-00.

6.2.3.2 Un-sequenced Data Packets

The Un-sequenced Data Packets act as an envelope to carry the actual data messages that are transferred from the client to the server. These messages are not sequenced and may be lost in the event of a socket failure. The higher-level protocol must be able to handle these lost messages in the case of a TCP/IP socket connection failure.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"U"	Un-sequenced Data Packet
Message	3	*	Alpha-numeric	Defined by a higher-level protocol, but must not contain any embedded linefeeds.

* = Variable values

6.2.3.3 Client Heartbeat Packets

The client should send a Client Heartbeat Packet anytime more than 1 second passes where no data has been sent to the server. The server can then assume that the link is lost if it does not receive anything for an extended period of time.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	"R"	Client Heartbeat Packet

6.2.3.4 **Logout Request Packets**

The client may send a Logout Request Packet to request the connection be terminated. Upon receiving a Logout Request Packet, the server will immediately terminate the connection and close the associated TCP/IP socket.

Field Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet.
Packet Type	2	1	“O”	Logout Request Packet