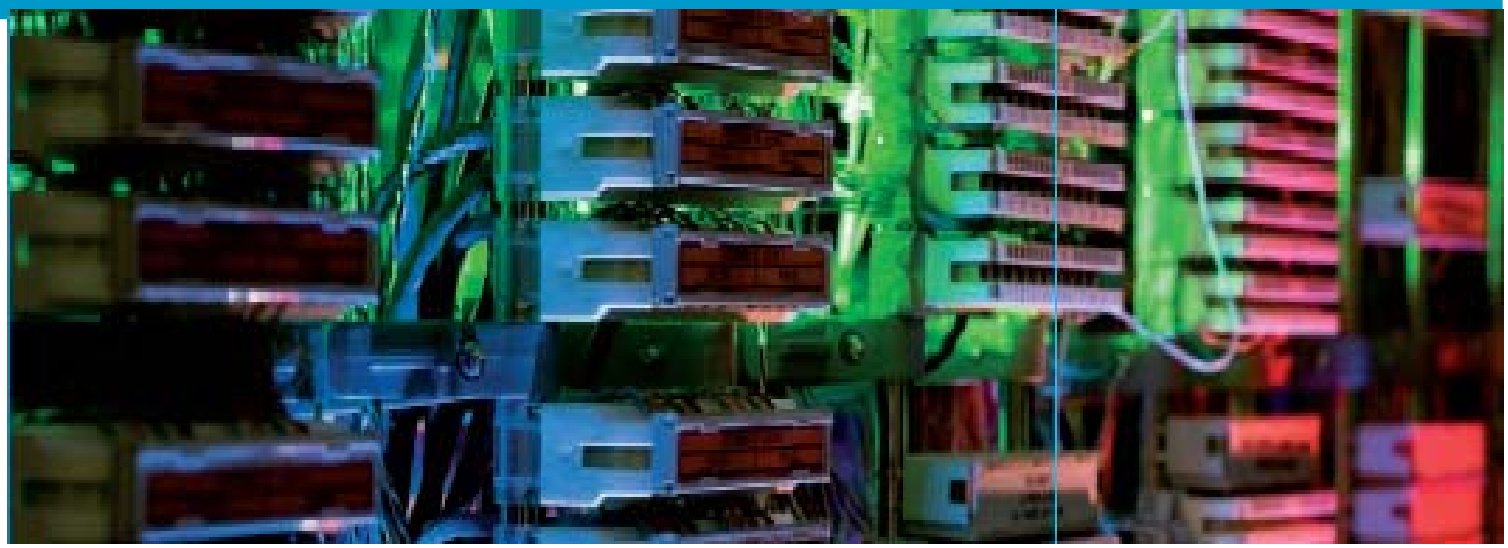
ITCH - Glimpse

Message Specification

ASX Market Information



Updated: 23 Mar 2012
Version: 1.0

Legal Notice

The information provided in this document and attachments does not represent ASX's or NASDAQ OMX's final technical design for the NASDAQ OMX Genium® INET API. ASX and NASDAQ OMX may make changes to the document and attachments during system development and testing which will be notified to users at a later date.

Whilst all reasonable care has been taken to ensure that the details are accurate and not misleading at the time of publication, no liability, to the extent permitted by law, (including liability to any person by reason of negligence), will be assumed by ASX or any subsidiary or employee of ASX, for any direct or indirect loss or damage caused by omissions from or inaccuracies in this document or otherwise from any use of the information contained in this document.

ASX and NASDAQ OMX reserve the right to change details in this document at any time without notice.

ASX Limited
ABN 98 008 624 691
20 Bridge Street, Sydney NSW 2000
PO Box H224
Australia Square NSW 1215

Telephone: 131 279
(+61 2) 9338 0000 (from overseas)
Email: info@asx.com.au

Contents

1 Introduction

ASX ITCH™ is the premium ultra-low latency 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 so meet the requirements of latency sensitive traders.

The ASX ITCH protocol provides:

- Ultra-low latency market information access (post full acceleration up to 7 times faster than current LCC's).
- Improved latency stability (reduced jitter).
- Control of the Socket.
- Multicast stream of order book changes.
- Full order detail, meaning every quote and every order in every ASX lit order book (ASX TradeMatch and PureMatch).
- Trade data for all ASX order books (ASX TradeMatch, VolumeMatch® , 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.

1.1 ITCH

ITCH is a direct data feed product, which features the following data elements:

- **Order level data (MBO) with broker ID**: The system will provide its full order depth using the standard ITCH format. ITCH uses a series of order messages to track the life of a customer order[1]. The ITCH message displays broker ID for non-anonymous instruments.

- **Trade messages**: 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.

[1] ITCH is an outbound market data feed only. The ITCH protocol does not support order entry.

- **Event controls**, such as the states of the different order books.
    - Order Book State message to inform receivers of state changes.

1.2 Glimpse

A complement to 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 the ASX Trade system. Glimpse uses the same message formats as ITCH.

Glimpse can be used to quickly sync up with the 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 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 ITCH sequence number to use when connecting to the real-time ITCH feed.

2 ITCH

2.1 Architecture

The 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 ITCH protocol are typically delivered using a higher level protocol that takes care of sequencing and delivery guarantees.

2.1.1 Protocol

The 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.

2.3 Message Formats

The 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 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 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 securities.

NOTE: Intra-day transmissions of this message will occur when new order books (securities) 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.

Name	Length	Value	Notes
Symbol	32	Alpha	The unique 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.
Number of decimals in Price	2	Numeric	This value defines the number of decimals used in price for this order book.
Number of decimals in Nominal Value	2	Numeric	This value defines the number of decimals used in 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.

Order Book Directory example message

Shows the Order Book ID (3104bc88) and other attributes for XJO11SEP4550EC.J88 for current day:

R,840622,3104bc88,XJO11SEP4550EC.J88,SEP-11 CALL OPT 4550 [XJOJ88],AU000XJOJ888,1,AUD,1,0,0,1,0,0

2.4.2 Combination Order Book Directory

The Combination Order Book Directory is a specialized directory message used for combinations. It represents both standard combinations defined by the exchange, and tailor-made combinations created by Participants.

Note: Intra-day transmissions of this message will occur when new combination order books are added to the system. 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 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.
Number of decimals in Price	2	Numeric	This value defines the number of decimals used in price for this order book.
Number of decimals in Nominal Value	2	Numeric	This value defines the number of decimals used in 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.

Name	Length	Value	Notes
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
Shows the individual legs that comprise the standard combination WAW11SEP12JAN_TF2:
M,604644,300058c3,WAW11SEP12JAN_TF2,,11,AUD,1,0,0,1,0,0,WAW11SEPF.1U,C,1,
WAW12JANF.2F,B,1,,?,0,,?,0

Shows the individual legs that comprise a tailor-made combination for BHP:
M,493439,14d6ffff,TMC_BHP_D_001,,,11,AUD,1,0,0,1,0,0,BHP,B,1,
BHP12DEC3456C,C,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.
Tick Size	8	Numeric	Tick Size for the given price range.
Price From	4	Price	Start of price range for this entry.
Price To	4	Price	End of price range for this entry.

Tick Size Table Entry example message
Shows tick size table for BHP:
L, 698551,BHP,1,1,99
L, 698551,BHP,5,100,1999
L, 698551,BHP,10,2000,999999900

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 as follows:

System Event Message				
Name	***Offset***	***Length***	***Value***	***Notes***
Message Type	0	1	"S"	System Event Message.
Event Code	1	1	Alpha	See System Event Codes below.

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

System Event Codes – Daily	
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 state changes.

Name	Length	Value	Notes
Message Type	1	"O"	Order Book State Message.
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	Order book identifier.
State Name	20	Alpha	Name of Order Book State.

Order Book State Change example message
Shows XJO11SEP4550EC.J88 moving to CLOSE:
O,040413, XJO11SEP4550EC.J88,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 with the correct 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.

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 order book. See *2.9 How to build an order book view* for details.

Name	Length	Value	Notes
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 display 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 **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 Participant ID example message
Shows a buy order entered in IRE for 876 @ 717 cents:
A, 845211,549f2944:005c8c1c,IRE,B,3,876,7170,0,2

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 unique 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 *2.9 How to build 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 display 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 **NOTE:** This field is a bit map. Multiple values may be set simultaneously.

Name	Length	Value	Notes
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
Shows a sell order entered by broker AU310 in XJO12MAR1500.EC for 35 @ 3 cents:
F, 313081,549f2943:005a8ae0,XJO12MAR1500EC.D27,S,1,35,30,0,2,AU310

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 for an order, ITCH subscribers must deduct the executed quantity stated in the Modify message from the original quantity stated in the Add Order message with the same Order ID. 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
Side	1	Alpha	The type of order being executed. Values: "B" = buy order. "S" = sell order.
Executed Quantity	8	Numeric	The number of shares executed.
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.

Name	Length	Value	Notes
Participant ID, counterparty	7	Alpha	Participant identifier of the counterparty to the execution. Blank for anonymous instruments.

Order Executed example message
Shows order ID 5314aec1:00007957 trading 100 lots:
E,420177,5314aec1:00007957,
XJO12MAR1500.EC,S,100,00000000:00d4afc1:00000009,AU550,AU551

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 participant is looking to use the ITCH data in trade tickers or volume calculations, ASX recommends that participants 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
Side	1	Alpha	The type of order being executed. Values: "B" = buy order. "S" = sell order.
Executed Quantity	8	Numeric	The number of shares executed.
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 of the executed order.
Occurred at Cross	1	Alpha	Values: "Y" = Yes, trade occurred in an auction "N" = No, trade occurred in continuous matching

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

Order Executed at Price example message
Shows order 549e7402:000fe88c trading 100 lots at price $31.25:
C,029192,549e7402:000fe88c,ASX,S,100,00d8c5c2:00000003:00000002,,,31250,Y,Y

2.6.2.3 Order Replace Message

This message is sent whenever an order in the book has been replaced. The remaining quantity from the original order is no longer accessible, and must be removed.

The Side, Order Book ID, and Participant Identifier (if non-anonymous) remain the same as the original order. Participant 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
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 order book. See *2.9 How to build an order book view* for details.
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
Shows sell order for BHP replaced for 1,000 @ 3570 cents at new position 7:
U,835289,54ad5081:00011f68,BHP,S,7,1000,35700,0

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 2: Order Delete messages are sent out when orders are inactivated. When central inactive orders are reactivated by a trading participant, 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	The Order book ID.
Side	1	Alpha	The type of order being deleted. Values: "B" = buy order. "S" = sell order.

Order Delete example message
Shows buy order in RIO being deleted:
D,029597,54ad5081:00011f6b,RIO,B

2.7 Trade Messages

2.7.1 Trade Message

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 participants 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
Quantity	8	Numeric	The quantity being matched in this execution.
Order Book ID	4	Numeric	Order Book Identifier.
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.
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
Shows execution for QAN in Centre Point for 2 @ 173.7 cents
P,381984,00d8c5c3:00000001:00000001,,2,QAN,1737,,,Y,N

2.8 Auction Messages

2.8.1 Equilibrium Price Update

This message is used when auctions occur. The message provides the changes in equilibrium 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.

Name	Length	Value	Notes
Timestamp – Nanoseconds	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	4	Numeric	The Order Book ID.
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
Shows BHP in PRE_OPEN state, 30 executable bids and 12 executable asks, (surplus volume is 18), equilibrium price is $44.00, best bid quantity is 30 @ $44.00, best ask quantity is 6 @ $43.90:
Z,909219,BHP,30,12,44000,44000,43900,30,6

Shows BHP moving to OPEN state:
O,800972,BHP,OPEN

Shows that the auction has occurred and an equilibrium price no longer exists:
Z,800972,BHP,0,0,-2147483648,-2147483648,-2147483648,0,0

2.9 How to build an order book view

The information needed to build an order book view from the ITCH message flow is contained in sections *2.6.1 Add Order Messages* and *2.6.2 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 shall be ranked by:

1. Price
2. 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 (with Price) message signals a partial or full fill. The order quantity must be reduced by the quantity of the Order Executed 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 How to build a Trade Ticker

The Trade Ticker is based on the following messages:

- Order Executed
- Order Executed with Price
- Trade

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

NOTE: Trade Reports are not included in ITCH.

3 Glimpse

3.1 Architecture

The Glimpse 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: Users must log in with SoupBinTCP sequence number 1 to correctly receive data.

3.1.1 Protocol

Protocol Option	Description
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 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. The sequence numbers are implicit, meaning that the client maintains a counter that is increased every time a message is received. At reconnect after a connection loss, the client submits the last seen sequence number in its Logon message, and the server resends every message starting from that sequence number. Note: Please refer to Nasdaq website for latest SoupBinTCP manual.

3.2 Data types

Glimpse messages have the same data types as ITCH messages.

3.3 Glimpse Messages

Glimpse uses a subset of ITCH messages.

Glimpse utilises all messages used by ITCH to describe the current state of the book, with the exception of those messages that change the state 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 ITCH sequence number to be used when connecting to the ITCH feed.

To maintain a real-time order display, real-time processing of 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	ITCH sequence number when the snapshot was taken. To be used when applying messages received via the ITCH MoldUDP feed. **NOTE:** While Glimpse is a binary feed, the SoupBinTCP uses ASCII characters to represent the sequence number.

4 Configuration and Infrastructure

4.1 Connectivity Requirements

4.1.1 Overview of Feeds

The following provides an overview of the ASX ITCH multicast feed components:

1. Each ASX switched environment provides dual redundant ITCH data services.
2. Each data service consists of:
    a. A multicast base address and base port for real time data dissemination using MoldUDP64 protocol.
    b. An IP address and base port for snapshot retrieval using SoupTCP protocol.
    c. An IP address and base port for MoldUDP64 retransmit requests.
3. Each base address or port requires the addition of the matching engine partition number to determine the actual address or port on which to receive the ITCH data. Eg multicast base address 233.54.12.231 port 21000 yields:
    a. Address 233.54.12.232:21001 for partition 1 data
    b. Address 233.54.12.233:21002 for partition 2 data
    c. (etc.)
    d. ASX Trade currently has five matching engine partitions.
4. Upon completion of the snapshot retrieval (Glimpse) an “end of snapshot” message will be received containing the ITCH sequence number with which to begin processing real time data.
5. It is mandatory to begin each day’s session with a Glimpse snapshot retrieval as these messages contain directory information in addition to the current order book snapshot.
6. The MoldUDP64 messages sent by the retransmit server as the result of a retransmit request will be received by the sending application on the port from which the request was sent.

4.1.2 Feed Address Examples

The following table gives examples of the various base address and the actual addresses which would be used for the matching engine partitions. Note these are from the Functional Test Environment and are different from the actual Production addresses, which will be provided by Market Access. The table shows the first three partitions; currently production actually has five partitions:

ITCH Feed A		ITCH Feed B	
MoldUDP64 Base Address and port			
233.54.12.223 21000 (Multicast UDP)		233.54.12.239 21100 (Multicast UDP)	
Partition 1	233.54.12.224:21001	Partition 1	233.54.12.240:21101
Partition 2	233.54.12.225:21002	Partition 2	233.54.12.241:21102
Partition 3	233.54.12.226:21003	Partition 3	233.54.12.242:21103
Glimpse IP Address and port (TCP)			
203.0.119.228 21800		203.0.119.229 21800	
Partition 1	203.0.119.228:21801	Partition 1	203.0.119.229:21801
Partition 2	203.0.119.228:21802	Partition 2	203.0.119.229:21802
Partition 3	203.0.119.228:21803	Partition 3	203.0.119.229:21803
Retransmit IP Address and port (UDP) – response received on sending port			
203.0.119.228 24000		203.0.119.229 24000	
Partition 1	203.0.119.228:24001	Partition 1	203.0.119.229:24001
Partition 2	203.0.119.228:24002	Partition 2	203.0.119.229:24002
Partition 3	203.0.119.228:24003	Partition 3	203.0.119.229:24003

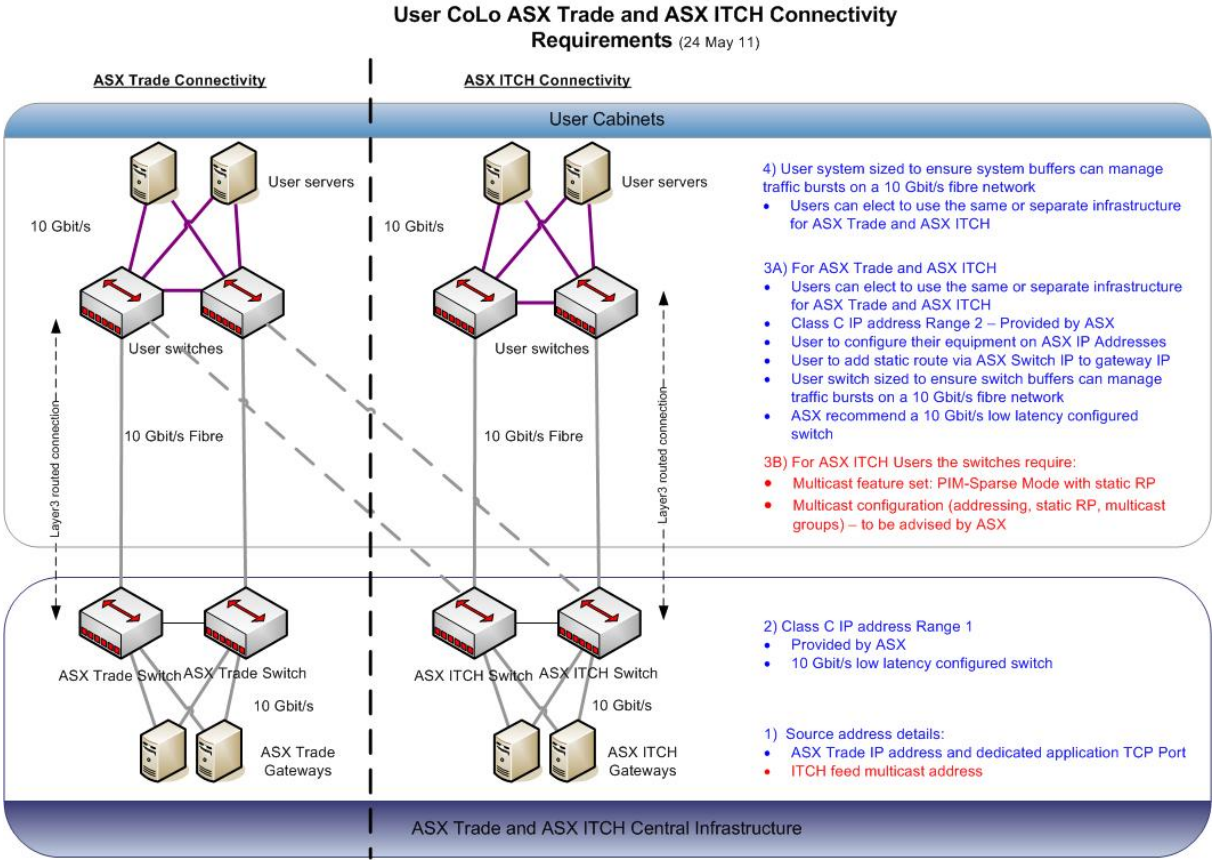
4.1.3 Configuration Details

ASX ITCH utilises multicast through switched infrastructure. Users of ASX ITCH will be required to allow for the following infrastructure requirements:

1. ASX will provide an ASX ITCH feed multicast address.
2. ASX will provide the specific Class C IP address range for the ASX infrastructure.
    a. ASX infrastructure will be on a 10 Gbit/s low latency configured network
3. ASX will provide another specific Class C IP address range for User switches/servers
    a. User to configure their equipment on the ASX provided Class C address range
    b. User to add static route on their equipment to provide a path to the IP address for each LCC provided in step 1

c. User switch infrastructure sized to ensure buffers can manage bursts on 10 Gbit/s network
d. Users can elect to use the same of separate infrastructure for access ASX ITCH
e. ITCH facing switches require a multicast feature set: PIM-Spare Mode with static RP
f. Multicast configuration (addressing, static RP, multicast groups) to be advised by ASX
g. ASX Cross Connects will be 10 Gbit/s Ethernet supplied on OM-3 multimode fibre pair with LC connectors. It is the Users responsibility to provide SFP's/XFP's (short reach) to terminate the fibres on their switches.

4. User server infrastructure sized to ensure buffers can manage traffic bursts on 10 Gbit/s network



4.1.4 ITCH Configuration Requirements

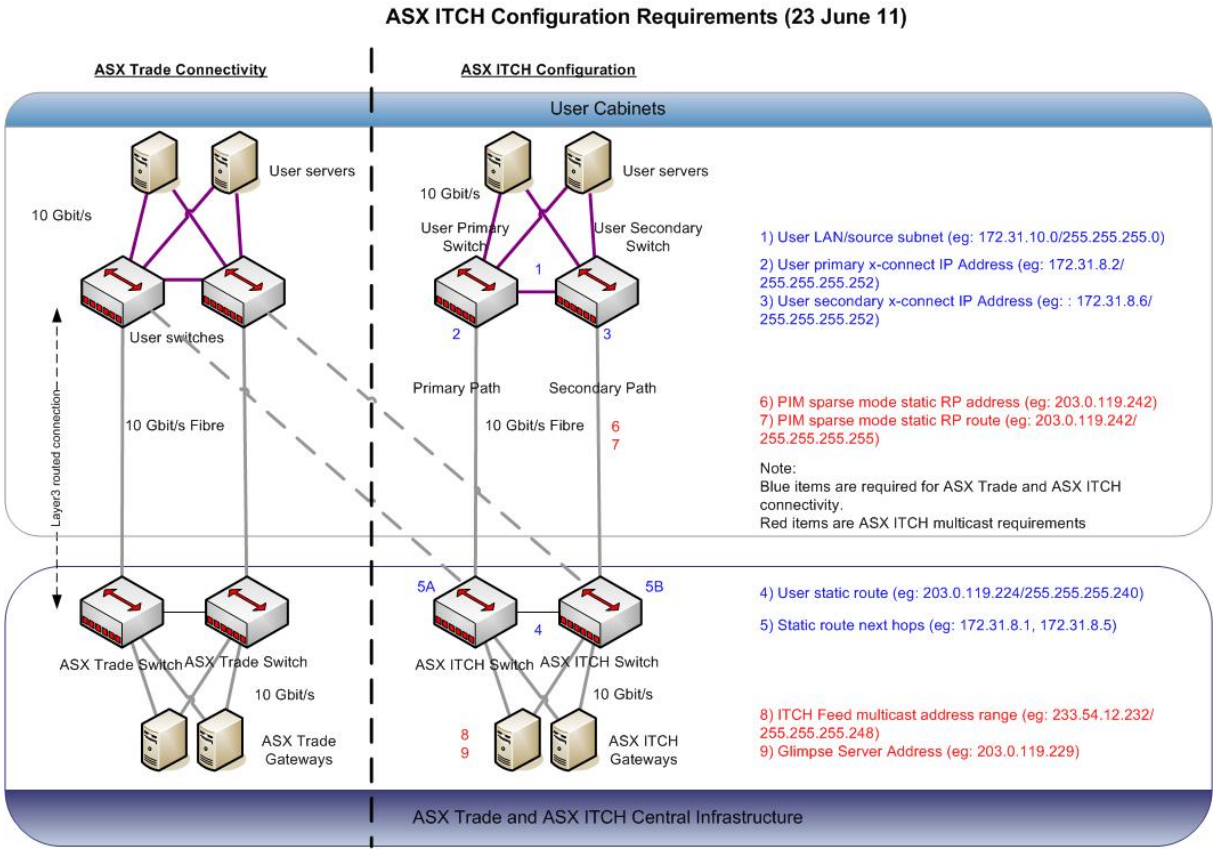
Users of ITCH will be required to configure for the below multicast solution. The following are the key elements for assessing the below specification:

- The ASX configuration was developed on Arista switches. Users will need to adjust the configuration for the devices they utilise.
- The ASX configuration has no access control applied – it is the responsibility of the user to manage their security requirements.
- The ASX configuration is engineered so that all traffic goes via the primary path in an active-standby arrangement. The second switch will only ever get used in the event of a network failure.
- PIM timers, PIM DR and IGMP timers have been tuned to obtain the fastest failover times in the event of a network failure within an ASX environment.

The ASX configuration is based on the following details:

1. User LAN/source subnet: 172.31.10.0/255.255.255.0
2. User primary cross connect IP address: 172.31.8.2/255.255.255.252
3. User standby cross connect IP address: 172.31.8.6/255.255.255.252
4. User static route: 203.0.119.224/255.255.255.240
5. Static route next hops: 172.31.8.1,172.31.8.5
6. PIM sparse mode static RP address: 203.0.119.242
7. PIM sparse mode static RP route: 203.0.119.242/255.255.255.255
8. ITCH feed multicast address range: 233.54.12.232/255.255.255.248 (this address range allocation will be the same for all Users, however this specific address is an example only)
9. Glimpse server address: 203.0.119.229
10. Multicast static route example (note no next hop for multicast routes): route add -net 233.54.12.232 netmask 255.255.255.248 ethX

Note: the provided addresses are an example only and will differ per User and per ASX environment (Production will be different to PTE). Specific addresses will be provided by Market Access during the new order process for each ASX environment.



A sample configuration is provided below which is based on Arista switch infrastructure. ASX will provide a specific sample configuration with all the user parameters when a user places an order for ITCH connectivity.

Sample Configuration

User switch sample configuration is as per the below and configures two sample vLAN's (100 and 1001). The below text in red is to highlight the multicast configurations required for ASX ITCH.

Note: this information is supplied as an example by ASX. Users need to consult directly with their hardware supplier for configuration recommendations and to address configuration issues. Users should test configurations with the ASX Trade Participant Test Environment before implementing into production.

Users switch 1 (primary) sample configuration:

```
vlan 100,1001

ip routing

ip multicast-routing
ip mfib activity polling-interval 5

ip pim rp-address 203.0.119.242 233.54.12.232/29
```

```
interface Ethernet1
  description Connection between switch 1 and 2
  switchport access vlan 100

interface Ethernet24
  description Primary ASX cross connect
  switchport access vlan 1001
  spanning-tree portfast

interface Vlan100
  description Customer Servers
  ip address 172.31.10.1/24
  ip igmp query-max-response-time 40
  ip igmp query-interval 5
  ip pim sparse-mode
  ip pim query-interval 2
  ip pim dr-priority 15
  vrrp 1 priority 20
  vrrp 1 ip 172.31.10.3

interface Vlan1001
  description Primary ASX cross connect
  ip address 172.31.8.2/30
  ip pim sparse-mode

ip route 203.0.119.224/28 172.31.8.1
ip route 203.0.119.224/28 172.31.10.2 100
ip route 203.0.119.242/32 172.31.8.1
ip route 203.0.119.242/32 172.31.10.2 100
```

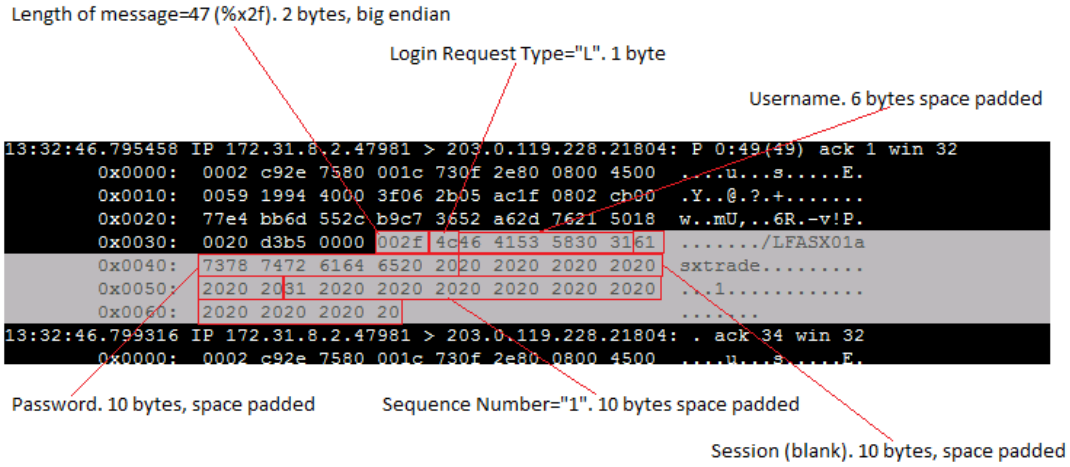
User switch 2 (standby) sample configuration:

```
vlan 100,1001

ip routing

ip multicast-routing
ip mfib activity polling-interval 5

ip pim rp-address 203.0.119.242 233.54.12.232/29

interface Ethernet1
  description Connection between switch 1 and 2
  switchport access vlan 100

interface Ethernet24
  description Secondary ASX cross connect
  switchport access vlan 1001
  spanning-tree portfast

interface Vlan100
  description Customer Servers
  ip address 172.31.10.2/24
  ip igmp query-max-response-time 40
  ip igmp query-interval 5
  ip pim sparse-mode
  ip pim query-interval 2
  vrrp 1 priority 10
  vrrp 1 ip 172.31.10.3
```

```
interface Vlan1001
  description Secondary ASX cross connect
  ip address 172.31.8.6/30
  ip pim sparse-mode

ip route 203.0.119.224/28 172.31.8.5
ip route 203.0.119.224/28 172.31.10.1 100
ip route 203.0.119.242/32 172.31.8.5
ip route 203.0.119.242/32 172.31.10.1 100
```

4.2 Glimpse Login

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



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

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.

5 Manual Updates History

Date	Bulletin	Page	Changes Made
Updates since manual version 1.0			

6 Support Contact Details

Contact the ASX Market Access support team either via email on MarketAccess@asx.com.au or phone 1800 663 053 (or on +612 9227 0372 from outside Australia).

After hours support for production problems is provided from 18:00 hours to 08:00 hours. Data content and test system problems are not supported during this period.

Contact: ASX Production Services – +61 2 9227 0821.

7 Appendix A: ITCH Scenarios

The following ITCH scenarios are presented:

Disclosed Orders, Continuous Trading

- New Order Entry, Amend and Delete
- New Order Fully Trades with Existing Order(s)
- New Order Partially Trades with Existing Order(s)
- Existing Order Amended and Fully Trades with Existing Order
- Existing Order Amended and Partially Trades with Existing Order

Undisclosed Order Executions, Continuous Trading

- Undisclosed order entered into book.
- New order fully trades with existing undisclosed order.
- New order fully trades with existing undisclosed order. Undisclosed order below undisclosed threshold.
- New order fully trades with existing undisclosed order. Undisclosed order empty.
- New order partially trades with existing undisclosed order.

Iceberg Order Execution, Continuous Trading

- New order fully trades with existing iceberg order. New order quantity less than iceberg shown order quantity.
- New order partially trades with existing iceberg order. New order quantity greater than, or equal to, iceberg shown order quantity, less than total quantity.
- New order partially trades with existing iceberg order. New order quantity greater than iceberg shown order quantity, greater than iceberg order total quantity.

Disclosed Order Execution, Auctions

- Orders trade at entered prices.
- Orders trade at different to entered prices.

Tailor-Made Combinations (TMCs)

- New TMC order trades with existing TMC order.
- New TMC order trades with existing orders in the book.
- Two TMC orders trade during an auction.

Instrument Creation

- TMC creation.
- Standard instrument creation.

Miscellaneous Tasks

- Participant to participant order transfer.

Note on Repeating Steps
A single asterisk (*) against a step indicates that the step may be repeated. Double asterisks (**) groups the blocks of steps that may repeat together.

7.1 Disclosed Orders, Continuous Trading

7.1.1 New Order Entry, Amend and Delete

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	A	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	Order Amended (A)	U	Order ID (A) qty and price (A)	*Modify quantity and price for Order ID*
3	Order Deleted (A)	D	Order ID (A)	*Remove Order ID (A)*

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	F	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	Order Amended (A)	U	Order ID (A) qty and price (A)	*Modify quantity and price for Order ID*
3	Order Deleted (A)	D	Order ID (A)	*Remove Order ID (A)*

7.1.2 New Order Fully Trades with Existing Order(s)

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	A	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2*	New order (B) fully trades with (A) (and possibly other existing orders)	E	Execution qty order ID (A) price (A)	*Subtract Executed quantity from stored Order ID (A)* *If stored Order ID (A) quantity is zero, delete the stored order*

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	F	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2*	New order (B) fully trades with (A) (and possibly other existing orders)	E	Execution qty order ID (A) price (A)	*Subtract Executed quantity from stored Order ID (A)* *If stored Order ID (A) quantity is zero, delete the stored order*

7.1.3 New Order Partially Trades with Existing Order(s)

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	A	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2*	New order (B) partially trades with (A) (and possibly other existing orders)	E	Execution qty order ID (A) price (A)	*Subtract Executed quantity from stored Order ID (A) If stored Order ID (A) quantity is zero, delete the stored order*
3		A	Residual Order ID (B) qty and price (B)	*Store Order ID, Quantity and Price for (B)*

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	F	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	New order (B) partially trades with (A) (and possibly other existing orders)	E	Execution qty order ID (A) price (A)	*Subtract Executed quantity from stored Order ID (A) If stored Order ID (A) quantity is zero, delete the stored order*
3		F	Residual Order ID (B) qty and price (B)	*Store Order ID, Quantity and Price for (B)*

7.1.4 Existing Order Amended and Fully Trades with Existing Order

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	A	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	Order Entered Into Book (B)	A	Order ID (B) qty and price (B).	*Store Order ID, Quantity and Price for (B)*
3	Order (B) Amended to trade with (A) (and possibly other existing orders)	D	Order ID (B)	*Remove Order ID (B)*
4**		E	Execution qty order ID (A)	*Subtract Executed quantity from stored Order ID (A). If stored Order ID (A) quantity is zero, delete the stored order*

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	F	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	Order Entered Into Book (B)	F	Order ID (B) qty and price (B).	*Store Order ID, Quantity and Price for (B)*
3	Order (B) Amended to trade with (A) (and possibly other existing orders)	D	Order ID (B)	*Remove Order ID (B)*
4**		E	Execution qty order ID (A)	*Subtract Executed quantity from stored Order ID (A). If stored Order ID (A) quantity is zero, delete the stored order*

7.1.5 Existing Order Amended and Partially Trades with Existing Order

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	A	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	Order Entered Into Book (B)	A	Order ID (B) qty and price (B).	*Store Order ID, Quantity and Price for (B)*
3	Order (B) Amended to trade with (A) (and possibly other existing orders)	D	Order ID (B)	*Remove Order ID (B)*
4**		E	Execution qty order ID (A)	*Subtract Executed quantity from stored Order ID (A). If stored Order ID (A) quantity is zero, delete the stored order*
5		A	Residual quantity (B) remaining qty and price	*Store Order ID (B) Quantity and Price*

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order Entered Into Book (A)	F	Order ID (A) qty and price (A).	*Store Order ID, Quantity and Price for (A)*
2	Order Entered Into Book (B)	F	Order ID (B) qty and price (B).	*Store Order ID, Quantity and Price for (B)*
3	Order (B) Amended to trade with (A) (and possibly other existing orders)	D	Order ID (B)	*Remove Order ID (B)*
4**		E	Execution qty order ID (A)	*Subtract Executed quantity from stored Order ID (A). If stored Order ID (A) quantity is zero, delete the stored*

				order
5		F	Residual quantity (B) remaining qty and price	*Store Order ID (B) Quantity and Price*

7.2 Undisclosed Order Executions, Continuous Trading

7.2.1 Undisclosed Order Entered Into Book

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Undisclosed Order Entered Into Book (A)	A	Order ID (A) qty 0 and price (A). Xtype=32	*Store Order ID, (unknown) Quantity and Price for (A)*

7.2.2 New Order Fully Trades with Existing Undisclosed Order

Step	Scenario – No MPID Attribution	ITCH Message	Attributes	Activity
1	Undisclosed Order Entered Into Book (A)	A	Order ID (A) qty 0 and price (A). Xtype=32	*Store Order ID, (unknown) Quantity and Price for (A)*
2*	New order (B) fully trades with (A) (and possibly other existing orders)	P	Execution qty and price	*Update trade volume only*

7.2.3 New Order Fully Trades with Existing Undisclosed Order. Undisclosed Order Below Undisclosed Threshold.

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Undisclosed Order Entered Into Book (A)	A	Order ID (A) qty 0 and price (A). Xtype=32	*Store Order ID, (unknown) Quantity and Price for (A)*
2*	New order (B) fully trades with (A) (and possibly other existing orders)	P	Execution qty and price	*Update trade volume only*
3		U	Order ID (A) remaining qty and price (A) Xtype=0	*Update stored Order ID (A) quantity, price and exchange order type.*

7.2.4 New Order Fully Trades with Existing Undisclosed Order. Undisclosed Order Empty.

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Undisclosed Order Entered Into Book (A)	A	Order ID (A) qty 0 and price (A). Xtype=32	*Store Order ID, (unknown) Quantity and Price for (A)*
2*	New order (B) fully trades with (A) (and possibly other existing orders)	P	Execution qty and price	*Update trade volume only*
3		D	Order ID (A)	*Delete the stored order (A)*

7.2.5 New Order Partially Trades with Existing Undisclosed Order

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Undisclosed Order Entered Into Book (A)	A	Order ID (A) qty 0 and price (A). Xtype=32	*Store Order ID, (unknown) Quantity and Price for (A)*
2*	New order (B) fully trades with (A) (and possibly other existing orders)	P	Execution qty and price	*Update trade volume only*
3		D	Order ID (A)	*Delete the stored order (A)*
4		A	Residual quantity (B) price (B)	*Store Order ID, Quantity and Price for (B)*

7.3 Iceberg Order Execution, Continuous Trading

7.3.1 New Order Fully Trades with Existing Iceberg Order. New Order Quantity less than Iceberg Shown Order Quantity.

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Iceberg Order Entered Into Book (A)	A	Order ID (A) shown qty and price (A).	*Store Order ID, Shown Quantity and Price for (A)*
2	New order (B) fully trades with (A)	E	Execution qty order ID (A) price (A)	*Subtract Executed quantity from stored Order ID (A).*

7.3.2 New Order Fully Trades with Existing Iceberg Order. New Order Quantity equal to Iceberg Shown Order Quantity

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Iceberg Order Entered Into Book (A)	A	Order ID (A) shown qty and price (A).	*Store Order ID, Shown Quantity and Price for (A)*
2	New order (B) fully trades with (A)	D	Order ID (A)	*Delete the stored order (A)*
3	New order (B) fully trades with (A)	P	Execution qty and price	*Update trade volume only. Iceberg shown quantity remains in book.*
4	Refreshed iceberg has changed position and/or quantity	A	Order ID (A) shown qty, price and new position (A).	*Refreshed Order ID, Shown Quantity, Price and Orderbook position for (A). Order ID will be the same as original (A) order.*

7.3.3 New Order Fully Trades with Existing Iceberg Order. New Order Quantity greater than or equal to Iceberg Shown Order Quantity. Refreshed Iceberg Shown Quantity changes Position in Orderbook

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Iceberg Order Entered Into Book (A)	A	Order ID (A) shown qty and price (A).	*Store Order ID, Shown Quantity and Price for (A)*
2	New order (B) fully trades with (A)	P	Execution qty and price	*Update trade volume only*
3*	New order (B) trades with other normal orders	E	Execution qty Order ID (other)	*Subtract Executed quantity from stored Order ID (other).*
4	Refreshed iceberg has changed position and/or quantity	U	Order ID (A) shown qty, price and position (A).	*Update Refreshed Order ID, Shown Quantity, Price and Orderbook position for (A). Order ID will be the same as original (A) order.*

7.3.4 New Order Fully Trades with Existing Iceberg Order. New Order Quantity greater than Iceberg Shown Order Quantity, less than Total Quantity. Refreshed Iceberg Shown Quantity less than Original Shown Quantity

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Iceberg Order Entered Into Book (A)	A	Order ID (A) shown qty and price (A).	*Store Order ID, Shown Quantity and Price for (A)*
2	New order (B) fully trades with (A). Reduces shown quantity of (A)	E	Execution qty order ID (A) original quantity less order ID (A) remaining quantity	*Subtract Executed quantity from stored Order ID (A).*
3		P	Executed qty of remainder of order (B)	*No change to order book. Volume is total traded volume less volume executed in step 2.*

7.3.5 New Order Partially Trades with Existing Iceberg Order. New Order Quantity greater than Iceberg Order Total Quantity

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Iceberg Order Entered Into Book (A)	A	Order ID (A) shown qty and price (A).	Store Order ID, Shown Quantity and Price for (A)
2	New order (B) partially trades with (A)	E	Execution qty order ID (A) price (A)	Subtract Executed quantity from stored Order ID (A).
3		P	Executed qty of remainder of order (A)	No change to orderbook.
4		A	Residual quantity (B) price (B)	Store Order ID, Quantity and Price for (B)

7.4 Disclosed Order Execution, Auctions

7.4.1 Orders Trade in Auction

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1*	Instrument Series Pre-Open	O	Instrument Series PRE_OPEN	Market is in pre-open. Indicates state change for each Instrument Series.
2	Order Entered Into Book (A)	A	Order ID (A) qty and price (A).	Store Order ID, Quantity and Price for (A)
3		Z	Equilibrium price message for stock if necessary.	Equilibrium message indicates auction price and surplus volume if applicable
4	Order Entered Into Book (B)	A	Order ID (B) qty and price (B).	Store Order ID, Quantity and Price for (B)
5		Z	Equilibrium price message for stock if necessary.	Equilibrium message indicates auction price and surplus volume if applicable
6*	Instrument Series Opened	O	Instrument Series OPEN	Market opens. Indicates state change for each Instrument Series.
7**		C	Execution qty order ID (A) price (A) Printable=Y (on Sell side)	Subtract Executed quantity from stored Order ID (A). If stored Order ID (A) quantity is zero, delete the stored order
8**		C	Execution qty order ID (B) price (A) Printable=N (on Buy side)	Subtract Executed quantity from stored Order ID (B). If stored Order ID (B) quantity is zero, delete the stored order

7.5 Tailor Made Combinations

7.5.1 New TMC Order trades with existing TMC Order

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order for TMC entered into market (TA)	F	Order ID (TA) qty and price (TA) of TMC.	*Store Order ID, Quantity and Price for (TA)*
2*	New order (TB) fully trades with (TA) (and possibly other existing orders for TMC)	C	Execution qty order ID (TA) price (TA) Printable=N	*Subtract Executed quantity from stored Order ID (TA) If stored Order ID (TA) quantity is zero, delete the stored order*
3*	Execution quantities for TMC legs are reported	P	Executed qty and price of legs 1-4 as required Printable=Y	*No change to order book. Update trade volume statistics.*

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order for TMC entered into market (TA)	A	Order ID (TA) qty and price (TA) of TMC.	*Store Order ID, Quantity and Price for (TA)*
2*	New order (TB) fully trades with (TA) (and possibly other existing orders for TMC)	C	Execution qty order ID (TA) price (TA) Printable=N	*Subtract Executed quantity from stored Order ID (TA) If stored Order ID (TA) quantity is zero, delete the stored order*
3*	Execution quantities for TMC legs are reported	P	Executed qty and price of legs 1-4 as required Printable=Y	*No change to order book. Update trade volume statistics.*

7.5.2 New TMC Order trades with existing Orders in the Book

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Order for TMC entered into market (TA)	F	Order ID (TA) qty and price (TA) of TMC.	Store Order ID, Quantity and Price for (TA)
2*	New order(s) (B…) entered against TMC leg(s)	F	Order ID (B…) qty and price (B…) of order	Store Order ID, Quantity and Price for (B…)
3	Bait generated by system for remaining leg	F	Order ID (C) qty and price (C) of bait.	Store Order ID, Quantity and Price for (C).
4	Order entered trades with bait order	E	Order ID (C) qty and price (C) of order	Subtract Executed quantity from stored Order ID (C)
5*	Order(s) (B…) trades against TMC leg	E	Order ID (B…) qty and price (B…) of order	Subtract Executed quantity from stored Order ID (B…)
6	Quantity of order against TMC is reduced	C	Execution qty order ID (TA) price (TA) Printable=N	Subtract Executed quantity from stored Order ID (TA) If stored Order ID (TA) quantity is zero, delete the stored order

Step	Scenario – No Participant ID	ITCH Message	Attributes	Activity
1	Order for TMC entered into market (TA)	A	Order ID (TA) qty and price (TA) of TMC.	Store Order ID, Quantity and Price for (TA)
2*	New order(s) (B…) entered against TMC leg(s)	A	Order ID (B…) qty and price (B…) of order	Store Order ID, Quantity and Price for (B…)
3	Bait generated by system for remaining leg	A	Order ID (C) qty and price (C) of bait.	Store Order ID, Quantity and Price for (C).
4	Order entered trades with bait order	E	Order ID (C) qty and price (C) of order	Subtract Executed quantity from stored Order ID (C)

5*	Order(s) (B…) trades against TMC leg	E	Order ID (B…) qty and price (B…) of order	Subtract Executed quantity from stored Order ID (B…)
6	Quantity of order against TMC is reduced	C	Execution qty order ID (TA) price (TA) Printable=N	Subtract Executed quantity from stored Order ID (TA) If stored Order ID (TA) quantity is zero, delete the stored order

7.5.3 Two TMC Orders trades during Auction

Step	Scenario – With Participant ID	ITCH Message	Attributes	Activity
1	Instrument Series Pre-Open	O	Instrument Series PRE_OPEN	Market is in pre-open. Indicates state change for each Instrument Series.
2	Order for TMC entered into market (TA)	F	Order ID (TA) qty and price (TA) of TMC.	*Store Order ID, Quantity and Price for (TA)*
3		Z	Equilibrium price message for stock if necessary.	Equilibrium message indicates auction price and surplus volume if applicable
4	Order for TMC entered into market (TB)	F	Order ID (TB) qty and price (TB) of TMC.	*Store Order ID, Quantity and Price for (TB)*
5		Z	Equilibrium price message for stock if necessary.	Equilibrium message indicates auction price and surplus volume if applicable
6	Instrument Series Opened	O	Instrument Series OPEN	Market opens. Indicates state change for each Instrument Series.
7**		C	Execution qty order ID (TA)	*Subtract Executed quantity from*

			price (TA) Printable=Y (on Sell side)	*stored Order ID (TA) If stored Order ID (TA) quantity is zero, delete the stored order*
8**		C	Execution qty order ID (TB) price (TB) Printable=N (on Buy side)	*Subtract Executed quantity from stored Order ID (TB) If stored Order ID (TB) quantity is zero, delete the stored order*
9*	Execution quantities for TMC legs are reported	P	Executed qty and price of legs 1-4 as required Printable=Y Occurred at Cross = N (TMC trades do not indicate that they resulted from an auction)	

7.6 Instrument Creation

7.6.1 TMC Creation

Step	Scenario	ITCH Message	Attributes	Activity
1	TMC Instrument Created	M	Combination Name TMC_XXX_X_NNN	Combination Name, Order Book ID and Leg Details provided
2*		L	TMC_XXX_X_NNN tick table entry	The tick table definitions for the instrument.
3		O	Instrument Series State	Indicates current state for Instrument Series.

7.6.2 Standard Instrument Creation

Step	Scenario	Tag	Attributes	Activity
1	Instrument Created	R	Instrument name XXX	Instrument Name, Order Book ID
2*		L	XXX tick table entry	The tick table definition for the instrument. This may be repeated.
3		O	Instrument Series State	Indicates current state for Instrument Series.

7.7 Miscellaneous Tasks

7.7.1 Participant to Participant Order Transfer

Step	Scenario – No Participant ID	ITCHTag	Attributes	Activity
1	Order Entered Into Book (A)	A	Order ID (A) Order book position (1).	*Store Order ID, Order book position (1)*
2	User to User Order Transfer from MMM to NNN	D	Order ID (A)	*Delete the Order from order book*
3		A	Order ID (A) Order book position (1).	*Store Order ID, & Order book position (1)*

Step	Scenario – With Participant ID	ITCHTag	Attributes	Activity
1	Order Entered Into Book (A)	F	Order ID (A) Participant ID (MMM) & Order book position (1).	*Store Order ID, Participant ID & Order book position (1) for (MMM)*
2	User to User Order Transfer from MMM to NNN	D	Order ID (A)	*Delete the Order for MMM from order book*
3		F	Order ID (A) Participant ID (NNN) & Order book position (1).	*Store Order ID, Participant ID (NNN) & Order book position (1)*

8 Appendix B: ITCH – Known Issues

ASX would like to bring to your attention a number of minor know issues. These will be addressed by ASX in subsequent releases of the software.

Details of the issues are as follows.

8.1 ASX Defect Code: 1915

ITCH: Duplicate O messages

For certain instruments, O messages are sent out twice at different times for the same session states.

For example, O messages OPEN for certain agricultural products are sent out once at 09:50, and then once more at 11:00. At 09:50 the instrument opens and at 11:00 the Agricultural Derivatives Market moves to OPEN.

O,Timestamp,54W12JUNF.2M,OPEN
O,Timestamp,54W12DECF.2Z,OPEN
O,Timestamp,54W12OCTF.2V,OPEN
O,Timestamp,54W12AUGF.2Q,OPEN
O,Timestamp,54W12FEBF.2G,OPEN
O,Timestamp,54W12APRF.2J,OPEN

O,Timestamp,54W12JUNF.2M,OPEN
O,Timestamp,54W12DECF.2Z,OPEN
O,Timestamp,54W12OCTF.2V,OPEN
O,Timestamp,54W12AUGF.2Q,OPEN
O,Timestamp,54W12FEBF.2G,OPEN
O,Timestamp,54W12APRF.2J,OPEN

This issue occurs for some products when both the Market and the Instrument Class or Instrument Type have a trading session schedule configured. In the above example, the market open time is different from the opening for Wool instrument classes.

The same also occurs for the Warrants Market, where Commodity and Currency Warrants open earlier than the main Warrants Market.

8.2 ASX Defect code: 1921

ITCH: Trade message dissemination for Warrant (or Derivatives) auction

Scenario 1: Orders executed with the same price in auction

STEPS:
1. All markets are in OPEN.
2. The underlying AMP goes into PRE_NR due to receipt of a sensitive company announcement.
3. Enter two orders for Warrant AMPIRA: Buy Qty 150 @ 160.0 cents, Sell Qty 150 @ 160.0 cents.
4. AMP resumes, an auction takes place.

RESULT:
E (Order Executed) messages are disseminated for orders matching, instead of C (Order Executed with Price) messages; State Change message O is received before the Execution messages.

O,Timestamp,AMPIRA,OPEN
Z,Timestamp,AMPIRA,0,0,-2147483648,-2147483648,-2147483648,0,0
E,Timestamp,OrderID1,AMPIRA,B,150,MatchID,PartID,PartID
E,Timestamp,OrderID2,AMPIRA,S,150,MatchID,PartID,PartID

EXPECTED RESULT:
1) C messages should be received for orders traded out in an auction.
2) The Uncross flags in the received C messages should be Y for trades from auctions.
3) The Printable flags in the received C messages for the traded Buy orders should be N, so the orders executed in the auction are not double-counted.
4) State Change message O should come after the execution messages, not before.

Scenario 2: Orders executed with a different price in auction

STEPS:
Repeat steps from Scenario 1, but instead use these orders in Step 3:

Order 1: Buy Qty 50 @ 170.0 cents
Order 2: Buy Qty 50 @ 165.0 cents
Order 3: Sell Qty 50 @ 160.0 cents
Order 4: Sell Qty 50 @ 155.0 cents

RESULT:
All orders are executed at price 165.0 in the auction. Orders entered at other prices (Orders 1, 3 and 4) will generate C messages on execution; orders (Order 2) entered at the same price will generate E messages.

O,Timestamp,AMPIRA,OPEN
Z,Timestamp,AMPIRA,0,0,-2147483648,-2147483648,-2147483648,0,0
C,Timestamp,OrderID1,AMPIRA,B,50,MatchID1,PartID,PartID,1650,N,Y
E,Timestamp,OrderID2,AMPIRA,B,50,MatchID2,PartID,PartID
C,Timestamp,OrderID3,AMPIRA,S,50,MatchID2,PartID,PartID,1650,N,Y
C,Timestamp,OrderID4,AMPIRA,S,50,MatchID1,PartID,PartID,1650,N,Y

EXPECTED RESULT:
1) C messages should be received for all orders traded out in an auction.
2) The Uncross flags in the received C messages should be Y for all trades from auctions.
3) The Printable flags in the received C messages for the traded Buy orders should be N, so the orders executed in the auction are not double-counted.
4) State Change message O should come after the execution messages, not before.

8.3 ASX Defect Code: 1924

ITCH: Trade message dissemination for auction when Trading Session State moves out of PRE_CSPA

Scenario 1: Auction when Trading Session State moves out of PRE_CSPA

STEPS:
1. The Market is in PRE_CSPA.
2. Enter the following orders for an Equity:

Order 1: Buy Qty 100 @ 25 cents
Order 2: Buy Qty 140@ 25.5 cents
Order 3: Buy Qty 100@ 26 cents
Order 4: Sell Qty 120 @ 23 cents
Order 5: Sell Qty 100 @ 24 cents
Order 6: Sell Qty 100 @ 27 cents

3. Market moves into CSPA. Auction takes place at price = 25.5.

RESULT:
All orders are executed at price 25.5 in the auction. Orders entered at other prices (Orders 3, 4 and 5) will generate C messages on execution; orders (Order 2) entered at the same price will generate E messages.

O,Timestamp,AGK,CSPA
Z,Timestamp,AGK,0,0,-2147483648,-2147483648,-2147483648,0,0
C,Timestamp,OrderID4,AGK,S,100,MatchID1,,,255,N,Y
C,Timestamp,OrderID4,AGK,S,20,MatchID2,,,255,N,Y
C,Timestamp,OrderID3,AGK,B,100,MatchID1,,,255,N,Y
E,Timestamp,OrderID2,AGK,B,20,MatchID2,,
E,Timestamp,OrderID2,AGK,B,100,MatchID3,,
C,Timestamp,OrderID5,AGK,S,100,MatchID3,,,255,N,Y

EXPECTED RESULT:
1) C messages should be received for all orders traded out in an auction.
2) The Uncross flags in the received C messages should be Y for all trades from auctions.
3) The Printable flags in the received C messages for the traded Buy orders should be N, so the orders executed in the auction are not double-counted.
4) State Change message O should come after the execution messages, not before.

Scenario 2: Auction when Trading Session State moves out of PRE_OPEN

STEPS:
Repeat all steps in Scenario 1, when the market moves from PRE_OPEN to OPEN.

RESULT:
Same result as in Scenario 1.

Scenario 3: Auction when Instrument Session State of PRE_NR ends for Equity

NOTE: This scenario returns the correct, expected results.
(This scenario differs from defect 1921 in section 8.2 in that the executions occur for an Equity, not a Warrant).

STEPS:
1. All markets are in OPEN.
2. The underlying AGK goes into PRE_NR due to receipt of a sensitive company announcement.
3. Enter the following orders:

Order 1: Buy Qty 100 @ 25 cents
Order 2: Buy Qty 140@ 25.5 cents
Order 3: Buy Qty 100@ 26 cents
Order 4: Sell Qty 120 @ 23 cents
Order 5: Sell Qty 100 @ 24 cents
Order 6: Sell Qty 100 @ 27 cents

4. AGK resumes, an auction takes place.

RESULT:
Results are as expected, C messages with correct Uncross and Printable flags are received for all orders executed in the auction; State Change message O comes last.

C,Timestamp,OrderID4,AGK,S,100,MatchID1,,,255,Y,Y
C,Timestamp,OrderID4,AGK,S,20,MatchID2,,,255,Y,Y
C,Timestamp,OrderID3,AGK,B,100,MatchID1,,,255,Y,N
C,Timestamp,OrderID2,AGK,B,20,MatchID2,,,255,Y,N
C,Timestamp,OrderID2,AGK,B,100,MatchID3,,,255,Y,N
C,Timestamp,OrderID5,AGK,S,100,MatchID3,,,255,Y,Y
O,Timestamp,AGK,OPEN

9 Appendix C: MoldUDP64 Manual

ASX MoldUDP64 Protocol Specification

V 0.2

17/08/2011

Date	Author	Notes
11/2/04	SL	Initial version.
12/4/08	SL	Updated for 64-bit support
2/24/09	HT	Corrected minor error in documentation
7/7/09	SM	Clarified the data length field definition

9.1 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 Re-request Servers can be placed nearby remote receivers to reduce latency and bandwidth over WAN links.

This document 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 datastream on a network. The listeners must subscribe to this multicast group to receive the downstream data. One or more Re-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.

9.2 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.

9.3 Terms

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.

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.

9.4 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.

Header

Downstream Packet Header

Field Name	Offset	Len	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

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.

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 that a Message Count of zero denotes a heartbeat and that a Message Count of OxFFFF(hex, or 65535 in decimal) denotes end of session.

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.

Downstream Packet Message Block

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

* = Variable values

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.

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.

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.

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 while 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.

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 Re-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 users 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).

Request Packet

Field Name	Offset	Len	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.

Sequence Number

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

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.

9.5 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 might obey the following flowchart:

1. Open a UDP socket for the appropriate port and join the desired multicast group.
2. Examine the first received packet to determine the currently active session.
3. If the received session does not match the expected session, abort and report the error.
4. Examine the sequence number of the first recently received packet.
5. If the sequence number does not match the next expected sequence number, send a Request Packet to the Request Server with expected packet number. Wait for a new packet and return to step 4.
6. Process each of the received messages in the packet. If a Downstream Packet with the Message Count set to End of Session is received, handle the End of Session event.
7. Wait for a new packet and return to step 4.

10 Appendix D: SoupBinTCP30_ouch40 Manual

SoupBinTCP

Version 3.00

10.1 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 that 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.

10.1.1 SoupBinTCP Logical Packets

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

Each SoupBinTCP logical packet has:

A. 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).
B. a single byte header which indicates the packet type.
C. a variable length payload

Two Byte Packet Length	Packet Type	Variable-length payload

SoupBinTCP Logical Packet Structure

Notes:

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.

10.1.2 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 Unsequenced Data Packets at any time after the Login Accepted Packet is received. These messages may be lost during TCP/IP socket connection failures.

10.1.3 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.

10.1.4 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.

10.1.5 Data Types

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

10.2 SoupBinTCP Packet Types

10.2.1 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.

Debug Packet

Field Name	Offset	Len	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

10.2.2 Logical Packets Sent by a SoupBinTCP Server

10.2.2.1 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.

Login Accepted Packet

Field Name	Offset	Len	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.

10.2.2.2 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.

Login Reject Packet

Field Name	Offset	Len	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.

10.2.2.3 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.

Sequenced Data Packet

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

* = Variable values

10.2.2.4 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.

Server Heartbeat Packet

Field Name	Offset	Len	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.

10.2.2.5 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 user will no longer be able to reconnect to the current session.

End of Session Packet

Field Name	Offset	Len	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.

10.2.3 Logical Packets Sent by the SoupBinTCP Client

10.2.3.1 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	Len	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	Alphanumeric	Username
Password	9	10	Alphanumeric	Password
Requested Session	19	10	Alphanumeric	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.

10.2.3.2 Unsequenced Data Packets

The Unsequenced 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.

Unsequenced Data Packet

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

* = Variable values

10.2.3.3 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.

Client Heartbeat Packet

Field Name	Offset	Len	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.

10.2.3.4 Logout Request Packet

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.

Logout Request Packet

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

10.3 Current Restrictions

None known.

10.4 Revision History

5.1 Version 1.00 – 9/1/1999

5.1.1 Initial distribution.

5.2 Version 2.00 - 10/29/2001

5.2.1 Added Heart beats in both directions to remove the dependence on TCP/IP keep-alive to detect link failures. Server and client are now both guaranteed to send something (either data or heartbeat) at least once per second. This way, if you don't hear anything from the socket for several seconds, you can assume the socket is dead and close it.

5.2.2 Added Debug Messages because they are handy for debugging problems. An example is to have a server send a Debug Message upon accepting a connection identifying the name of the machine. This way, someone can TELNET into a server and immediately verify that they have the right host.

5.2.3 Added "envelopes" around both the outbound Sequenced Messages and the inbound Unsequenced Message to differentiate them from the heartbeats.

5.3 Version 3.00 - 10/15/2008

5.3.1 Widened Sequence Number fields on login related packets in order to support streams of 1 billion messages or larger.

5.3.2 Added an explicit End of Session Packet so that the SoupBinTCP session layer code can more cleanly detect the end of a SoupBinTCP session.

5.3.3 Initial distribution of SoupBinTCP3.0, taken from SoupTCP3.0. Packets are two byte length-prefixed instead of new-line terminated, but otherwise the packet types are the same.

5.4 Version 3.00 – 07/20/2009

5.4.1 Removed conflicting sentence from Sequenced data packet description