Derivatives Clearing System Open Interface

Messages1.5.0

3 March 2025

Document Change Summary

Date	Version	Section	Change Description
May18	1.4.8	2.1.13, 3	New message: SendGiveUpUndoRequest_V1
Dec24	1.5.0	2.1.9, 3	New message: SendEODRepRequest_V1
Feb25	1.5.0	2.1.1	New message SendAcc_V2

Contents

1. Messages Overview 5

1.1. Message Transfer 5
1.2. ActiveX Controls 6
1.3. SendMsg ActiveX Control 6
1.4. RecMsg ActiveX Control 7
1.5. RecMsg ActiveX Control Retrieval Methods 8

2. Messages Sent to and Received from ASXCL 10

2.1. Messages Sent to ASXCL 10
2.2. Messages Received from ASXCL 27

3. Method Argument Definitions 51

4. Error Handling 71

4.1. Error Handling in ActiveX Controls 71
4.2. Rejection Messages 71

5. Method to Message Type Cross Reference 77

6. Appendix 1 - Glossary 80

Derivatives Clearing System Open Interface

Messages

3 March 2025

Introduction

Derivatives Clearing System (DCS) is a clearing system for ASX Clear (ASXCL) Pty Limited (Derivatives). DCS contains a messaging interface that enables participants to:

- Receive derivatives trade messages
- Receive all end of day data including product reference data, margin and settlement information
- Send and receive messages allocating trades
- Give and take up positions
- Exercise positions.

This document assumes an understanding of ASXCL derivatives markets and the use of ActiveX controls.

For more information on the software configuration of DCS, see *Introduction to ASX Derivatives Clearing System Open Interface V1.5.0*.

Terminology

Within DCS, commonly used industry terminology has been adopted. However, it has been necessary to introduce certain new terms that have a particular meaning specifically in DCS. These terms are listed in the table below.

Term	Definition
Allocation	The designation of a Trade to an account of a participant.
Default Account	Each participant is required to maintain an account that is used to hold any contracts traded by or on behalf of the participant that was not allocated to an account or given-up to another participant. The Default Account for each participant is automatically established and is assigned an Account Code of “0000000000”. This code cannot be amended.
Derivative Product	The code associated with a contract specification traded on an exchange.
Give-Up	The act of one participant passing contracts to another participant so that the contract can be cleared by the other participant.
Member Reference Prices	Prices determined by a participant that are used by the automatic exercise process.
Reference Prices	Prices issued by ASXCL, which a participant may choose to use as the basis for determining whether options (expiring ‘today’) should be automatically exercised.
Take-Up	The acceptance of contracts that have been given up by another participant.
Trade	A transaction (for Derivative Product contracts) dealt on the current business day. Within DCS, each transaction executed on an exchange/market generates two trades (one for

	the buyer and one for the seller). A trade record is also created for a 'take-up' participant when a trade is given-up.
Traded Entity	A futures delivery month or option series.

1. Messages Overview

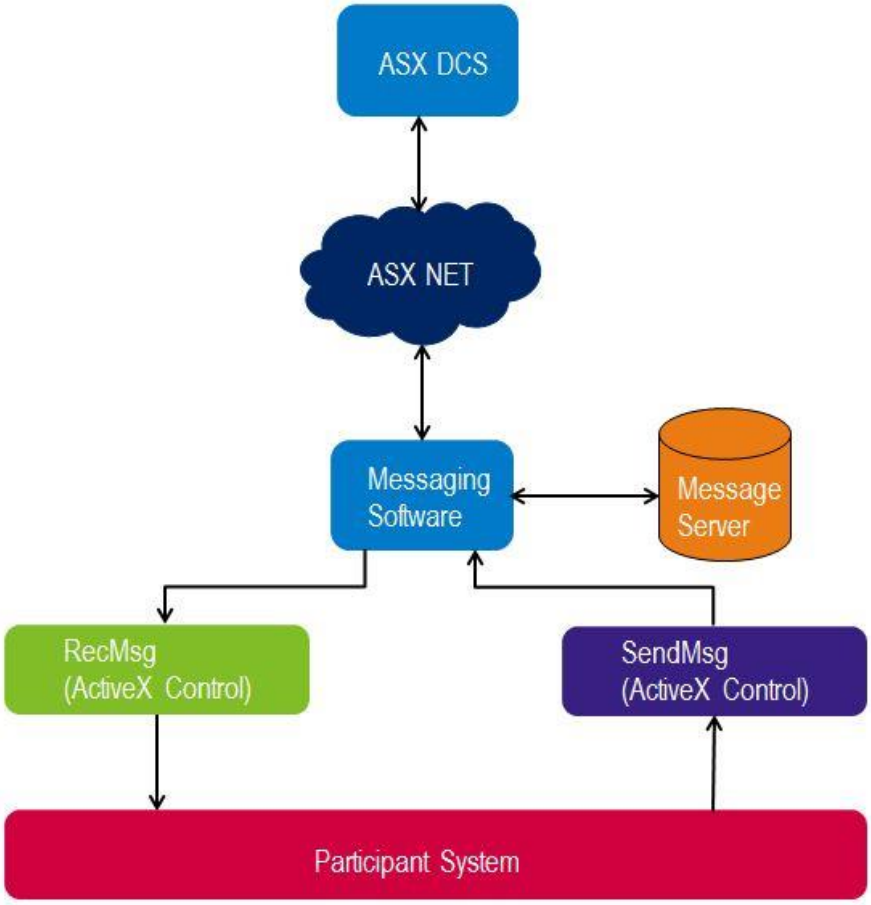
Messages are sent to and from the DCS to enable participants to manage accounts. This enables participants to receive derivatives trade messages, margin information and other market messages. DCS also sends and receives messages allocating trades, giving and taking-up positions and exercising positions.

1.1. Message Transfer

Two ActiveX controls (SendMsg and RecMsg) are supplied to participants to assist them in presenting the required data content of messages to be transmitted to ASXCL and for retrieving the data content of messages transmitted by ASXCL. The ActiveX controls interface to SendMsgH and RecMsgH that are implemented as Windows services. The SendMsgH and RecMsgH Windows services together with other software and a database (used for storing both messages sent and messages received) are also supplied by ASXCL to participants.

The transfer of all DCS messages must use the ActiveX controls and Windows services supplied by ASXCL. Participants should avoid direct access to the database used for storing messages.

The diagram below illustrates how a participant system interfaces to the messaging components.



1.2. ActiveX Controls

There is one method and two public properties that are common to both SendMsg and RecMsg ActiveX controls. This includes:

- **Method** - The Connect Method is used to establish a connection to the message server for use by the SendMsg and RecMsg ActiveX controls. The method returns a reference to the message database that is used for subsequent calls to send and receive messages.
- **Public Property** – Properties include:
- **ErrorNum** - This property contains the error number associated with an error encountered during a call to one of the methods associated with either the SendMsg or RecMsg ActiveX controls.
- **ErrorDesc** - This property contains a description of an error encountered during a call to one of the methods associated with either the SendMsg or RecMsg ActiveX controls.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgDatabaseName	In	Yes	NA
2	awrk_MsgWorkspace	In	Yes	NA

Special Considerations

The connect method should only be called once for each ActiveX control per session.

1.3. SendMsg ActiveX Control

The SendMsg ActiveX control contains one method for each valid message type that can be sent to ASXCL. Descriptions of these methods, how to use them and their associated arguments are all detailed within this document.

All SendMsg ActiveX methods return a Boolean value that indicates whether or not the message was accepted by the messaging software and has therefore been queued for transmission to ASXCL. If the method returns a value of False, then the contents of the public properties ErrorNum and ErrorDesc provide information on why the method failed.

Note:

It is the responsibility of the calling application to ensure that calls to methods of the SendMsg ActiveX control are encapsulated within a transaction along with all associated updates to the application database. This approach ensures that messages sent to ASXCL are consistent with the application database.

The following example illustrates the SendAcc_V1 method of the SendMsg ActiveX control.

```
lv_RetVal = SendMsg.SendAcc_V1(gs_UserId, Is_AuditType, ll_AccId, _Is_SegType,
ls_AutoMatchout, ls_AutoExer, ls_AccType, ls_Stat, ls_Acc, _ls_AccName,
_ls_SpecificCover, Is_MbrInfo, Is_Address1, Is_Address2, Is_Address3,
Is_Address4, Is_AccNameConfirm)
If Not lv_RetVal Then
   Workspaces(0).Rollback
              MsgBox "Error :- " & SendMsg.ErrorNum & "-" & SendMsg.ErrorDesc
              End
End If
```

1.4. RecMsg ActiveX Control

DCS uses two queues for each participant for the receipt of messages transmitted by ASXCL. One of these queues contains messages with a standard priority and the other queue contains messages with a high priority. Messages within each queue are held in chronological order. Messages should always be retrieved from the high priority queue ahead of messages in the standard priority queue. In response to a general request to 'get the next unread message' from a particular queue, unread messages are always retrieved in the chronological order in which they are received. However the RecMsg ActiveX control also provides a method for the retrieval of a specific message from a message queue.

There are three methods of the RecMsg ActiveX control that cater for the retrieval of messages (additional information is provided in the next section, *RecMsg ActiveX Control Retrieval Methods).*

This includes:

- GetMsg
- GetNextMsg
- Move ToNextMsg.

The GetNextMsg method of the RecMsg ActiveX control returns the details of the next message to be retrieved on a particular queue. The following example demonstrates how to use the GetNextMsg method of the RecMsg ActiveX control to obtain the next message to process. Note that the example first attempts to retrieve a message from the high priority message queue.

```
If Not RecMsg.GetNextMsg(gs_PriorityMsgQueue, ll_MsgSeq,  _
ls_MsgType, ll_MsgVersion, ll_MsgSetId, ls_MsgStartEnd, ls_MsgText) Then
If RecMsg.ErrorNum = 2004 Then
       'There are no messages to process
        …
Else
       'Error processing
         …
End If
Else
'Process the message
…
End If
```

If there is a 'next message' waiting, GetNextMsg returns the value 'True' and ls_MsgType contains a two character code identifying the type of message. If there is no 'next message' waiting, GetNextMsg returns a value of False, and ErrorNum is set to the value of 2004. If ErrorNum is set to a different value then the contents of the public property ErrorDesc should be interrogated to ascertain the cause of the error.

The following example (that continues on from the previous example) illustrates how the values returned by GetNextMsg can be examined to determine the type of message and version of that message to be read.

```
Else
           'Error processing
           …
End If
Else
   'Process the message here
Select Case ls_MsgType
Case 'TR'
          'Process trades
           …
Case 'GA'
             'Process the give-up advice
             'Obtain the parameters relating to the Give Up Advice message
              If Not RecMsg.GetGUAdvice_V1(ls_MsgText, ll_TrId, ll_AllocSeq,_
```

```
ls_AcceptFlag) Then
                          'Error processing
                           …
              End If
                 'Update the application database with the details
              …
Case 'TE'
     'Process Trade entity, first check the message version
     Select Case 11-MsgVersion
     Case 1
     'process GetTradedEntity_V1
     …
     Case 2
     'Process GetTradedEntity_V2
     …
     Case Else
     'Error unsupported message'
     End Select
Case …
Case Else
                    'Error unrecognised message type
                   …
End Select
End If
```

Once the data content of the message has been successfully processed, the MoveToNextMsg method of the RecMsg ActiveX control should be invoked. The MoveToNextMsg method causes the RecMsg ActiveX control to position itself to the next message within the message queue so that the next time the method GetNextMsg is called, the data content of that message is returned.

```
 'Move to the next message
If Not RecMsg.MoveToNextMsg(1s_MsgQueue, 1l_MsgSeq) Then
   'Error processing
   …
End If
```

Important

The MoveToNextMsg method should not be invoked until the current message is successfully finished processing, or precautions have been taken to ensure that the message content is safely stored so it can be subsequently retrieved.

1.5. RecMsg ActiveX Control Retrieval Methods

1.5.1 GetMsg

The GetMsg method is used to retrieve the contents of a specific message from one of the input queues. The input arguments are used to tell the ActiveX control the name of the message queue to obtain the message from, and the sequence number of the message that is required to be returned.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgQueue	In	Yes	NA
2	al_MsgSeq	In	Yes	NA
3	as_MsgType	Out	Yes	NA
4	al_MsgVersion	Out	Yes	NA
5	al_MsgSetID	Out	Yes	NA

6	as_MsgStartEnd	Out	Yes	NA
7	as_MsgText	Out	Yes	NA

1.5.2 GetNextMsg

The GetNextMsg method is used to return the next message from the nominated message queue. Where no more messages are queued the method returns 'False' and the ErrorNum property contains the value 2004.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgQueue	In	Yes	NA
2	al_MsgSeq	Out	Yes	NA
3	as_MsgType	Out	Yes	NA
4	al_MsgVersion	Out	Yes	NA
5	al_MsgSetID	Out	Yes	NA
6	as_MsgStartEnd	Out	Yes	NA
7	as_MsgText	Out	Yes	NA

1.5.3 Move ToNextMsg

The MovetoNextMsg method is used to move the pointer within the nominated queue to the nominated position. This method would be typically used to move to the next message after a message has been successfully processed.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgQueue	In	Yes	NA
2	al_MsgSeq	In	Yes	NA

2. Messages Sent to and Received from ASXCL

Messages are sent and received to and from ASXCL using method types. Messages can also be issued by ASXCL that are solely a message type, and do not have an associated method.

The following icons are used to indicate whether a message is sent to ASXCL or received from ASXCL.

Message is sent to ASXCL.

Message is received ASXCL.

2.1. Messages Sent to ASXCL

Messages sent to ASXCL include methods of the SendMsg ActiveX control and their arguments.

Note:

Where an argument is described as being optional, it indicates that the value is optional (i.e. depending on the data type values of 0 or "" are permissible). However the argument must always be provided. Arguments that are described as "In" must be supplied when activating the method, and arguments that are described as "Out" are those that are returned by the method. Also, note that the use of single quotation marks and tab characters is not permitted in any arguments.

For each method there is a two character Message Type. This is the code associated with a particular message, i.e. the value of ls_MsgType when using the GetNextMsg method. The Message Type is largely irrelevant for messages sent to ASXCL since it is automatically determined by the SendMsg ActiveX control, depending upon the method invoked.

2.1.1 SendAcc_V1, SendAcc_V2

Message Type - AC

The SendAcc_V1 method is used to transmit details of an account to ASXCL. These details are required when an account is first established and when an existing account is amended or deleted.

The SendAcc_V2 method is identical to SendAcc_v1 with an additional parameter for the account's equity clearer id.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_AmendmentType	In	Yes	
3	al_AccID	In	Yes	
4	as_SegType	In	Yes	Not required if as_AmendmentType = D
5	as_AutoMatchout	In	Yes	Not required if as_AmendmentType = D
6	as_AutoExer	In	Yes	Not required if as_AmendmentType = D
7	as_AccType	In	Yes	Not required if as_AmendmentType = D

8	as_AccStat	In	Yes	Not required if as_AmendmentType = D
9	as_Acc	In	Yes	Not required if as_AmendmentType = D
10	as_AccName	In	Yes	Not required if as_AmendmentType = D
11	as_SpecificCover	In	Yes	Not required if as_AmendmentType = D
12	as_MbrInfo	In	Yes	Not required if as_AmendmentType = D
13	as_Address1	In	Yes	Not required if as_AmendmentType = D
14	as_Address2	In	Yes	Not required if as_AmendmentType = D
15	as_Address3	In	Yes	Not required if as_AmendmentType = D
16	as_Address4	In	Yes	Not required if as_AmendmentType = D
17	as_AccNameConfirm	In	Yes	Not required if as_AmendmentType = D
18	as_Reason	In	Yes	Not required if as_AmendmentType = D
19	al_EquityID	In	V2 only	Not present in SendAcc_V1 SendAcc_V2: Not required if as_AmendmentType = D

Special Considerations

ASXCL only permits the deletion of an account under certain circumstances. This includes when:

The account cannot have an open position

The account cannot be the default account for the participant.

2.1.2 SendAccGrp_V1

Message Type - AG

The SendAccGrp_V1 method is used to transmit details of an account to ASXCL. These details are required when an account is first established and when an existing account is amended or deleted.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	

2	as_AmendmentType	In	Yes	
3	al_GrpID	In	Yes	
4	as_Group	In	Yes	Not required if as_AmendmentType = D
5	as_GrpName	In	Yes	Not required if as_AmendmentType = D

Special Considerations

The ASXCL only permits the deletion of an account group when the account group contains no accounts, i.e. no accounts are associated with the account group.

2.1.3 SendAccGrpMembership_V1

Message Type - AM

The SendAccGrpMembership_V1 method is used to transmit details of a change in the relationship between an account and an account group. The change is either the association of an account with an account group, or the deletion of the association between an account and an account group.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_AmendmentType	In	Yes	Only amendment ‘N’ew and ‘D’elete
3	al_GrpID	In	Yes	
4	Al_AccID	In	Yes	

2.1.4 SendAlloc_V1

Message Type - AL

The SendAlloc_V1 method is used to notify ASXCL that a trade has been partly or fully allocated to a nominated account.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_TrID	In	Yes	
4	al_AllocSeq	In	Yes	
5	al_Qty	In	Yes	The quantity is an absolute number.
6	al_AccID	In	Yes	
7	as_OpenClose	In	No	Reserved for future use.

8	as_AllocRef	In	No	
9	ac_Comm	In	No	
10	ac_CommBasisVal	In	No	
11	as_CommBasis	In	No	
12	as_ChargeGST	In	Yes	Can contain either a value of 'Y' or 'N'.

Special Considerations

SendAlloc_V1 cannot be used for amendments to allocations. Allocations must be reversed using the SendUndoAlloc_V1 and then resent using the SendAlloc_V1 method. Note that allocation sequence numbers should not be reused.

2.1.5 SendBCastViewed_V1

Message Type - BV

The SendBCastViewed_V1 method is used to notify ASXCL that a message broadcast has been viewed at the participant site.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_BCastID	In	Yes	

Special Considerations

This method should be sent to ASXCL when a new message broadcast is viewed for the first time.

2.1.6 SendCashWithdrawals_V1

Message Type - CW

The SendCashWithdrawals_V1 method is used to notify ASXCL of cash withdrawal details.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_Ledger	In	Yes	
3	as_SegType	In	Yes	
4	as_Cur	In	Yes	
5	ac_RequestedAmt	In	Yes	

Special Considerations
The cash withdrawal amount must not exceed the available amount the participant can withdraw. The amount must be greater than zero. If the amount is zero, then the DCS deletes any previously sent cash withdrawals. Cash withdrawals received after the cut-off times are rejected.

2.1.7 SendCHStatusCheckResponse_V1

Message Type - SM

The SendCHStatusCheckResponse_V1 method is used to respond to a Status Check message ('ping') sent by ASXCL. It enables ASXCL to determine whether or not communications with a participant site are functioning correctly.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	Al_StatusCheckSeq	In	Yes	
3	as_SegType	In	Yes	Only required by MCM installations.

Special Considerations
This method should be called after receipt of GetCHStatusCheck_V1 message from ASXCL. The message echoes the status check sequence number received from ASXCL.

2.1.8 SendCommRate_V1

Message Type - CR

The SendCommRate_V1 method is used to transmit details of default take-up and give-up commission rates to ASXCL. These details are required when commission rates are first established, and when existing rates are amended or deleted.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_AmendmentType	In	Yes	
3	al_OtherMbrID	In	Yes	.
4	as_DerivProdType	In	Yes	
5	as_Cur	In	Yes	

6	ac_GUComm	In	Yes	Not required if as_AmendmentType = D
7	as_GUCommBasis	In	Yes	Only Commission Basis types Rate and Percentage permitted.
8	ac_TUComm	In	Yes	Not required if as_AmendmentType = D
9	as_TUCommBasis	In	Yes	Only Commission Basis types Rate and Percentage permitted.

2.1.9 SendEODRepRequest_V1

Message Type - ER

The SendExerciseExclude_V1 method is used to notify ASXCL of open positions that expire at end of day and should be excluded from the automatic exercise process.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_RepID	In	Yes	
3	as_Frequency	In	Yes	.
4	as_Produce	In	Yes	
5	as_TonightOnly	In	Yes	

Special Considerations

This transaction has no effect until the commencement of the automatic exercise process during the end of day. Sending multiple SendExerciseExclude_V1 messages for the same open position simply overwrites the previous quantity to be excluded.

Note: ASXCL has the authority to override exercise exclusion instructions sent by a participant.

2.1.10 SendExerciseExclude_V1

Message Type - XX

The SendExerciseExclude_V1 method is used to notify ASXCL of open positions that expire at end of day and should be excluded from the automatic exercise process.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	

3	al_AccID	In	Yes	.
4	al_EntID	In	Yes	
5	al_Qty	In	Yes	This quantity is an absolute number.

Special Considerations

This transaction has no effect until the commencement of the automatic exercise process during the end of day. Sending multiple SendExerciseExclude_V1 messages for the same open position simply overwrites the previous quantity to be excluded.

Note: ASXCL has the authority to override exercise exclusion instructions sent by a participant.

2.1.11 SendExerciseManual_V1

Message Type - XE

The SendExerciseManual_V1 method is used to notify ASXCL of open positions that are to be manually exercised during the end of day.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_AccID	In	Yes	.
4	al_EntID	In	Yes	
5	al_Qty	In	Yes	This quantity is an absolute number.

Special Considerations

This transaction has no effect until the commencement of end of day processing by ASXCL. Sending multiple SendExerciseManual_V1 messages for the same open position simply overwrites the previous quantity to be exercised.

Note: ASXCL has the authority to override manual exercise instructions sent by a participant.

2.1.12 SendExerciseResponse_V1

Message Type - XR

The SendExerciseResponse_V1 method is used to acknowledge receipt of the exercise cut-off notification. The exercise cut-off is notified on the GetFacility_V1 message. The participant site must respond with a SendExerciseResponse_V1 when the as_Exercise argument on the facility message changes from 'Y' to 'N'.

Sequence	Argument	In/Out	Mandatory	Notes

1	as_UserID	In	Yes	

2.1.13 SendGiveUp_V1

Message Type - GU

The SendGiveUp_V1 method is used to notify ASXCL that a trade has been partly or fully given-up to a nominated participant.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_TrID	In	Yes	.
4	al_AllocSeq	In	Yes	
5	al_Qty	In	Yes	The quantity is an absolute number.
6	al_MbrFor	In	Yes	
7	ac_Comm	In	Yes	
8	ac_CommBasisVal	In	Yes	
9	as_CommBasis	In	Yes	
10	as_AllocRef	In	No	

2.1.14 SendGiveUpUndoRequest_V1

Message Type - GR

The SendGiveUpUndoRequest_V1 method is used to request ASXCL to undo a give-up.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_TrID	In	Yes	
4	al_AllocSeq	In	Yes	
5	as_RejReason	In	Yes	

Special Considerations

This is a request to undo the give-up. If the take-up participant has not accepted the give-up, the request will be accepted by ASXCL via a Give-up advice message (GetGUAdvice_V1) with accept flag (as_AcceptFlag) set to D i.e. deleted by ASXCL.

Special Considerations
Requests to undo a give-up can be made if the give-up has not been accepted or rejected by the other party.

2.1.15 SendMatchOut_V1

Message Type - MO

The SendMatchOut_V1 method is used to notify ASXCL of back-to-back open positions that are to be manually matched out during the end of day processing.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_AccID	In	Yes	.
4	al_EntID	In	Yes	
5	al_Qty	In	Yes	This quantity is an absolute number.

Special Considerations
There is no need to send match out details for accounts that are set to automatically match out back-to-back positions, as back-to-back positions are automatically matched out during end of day processing. This method has no effect until the commencement of end of day processing by ASXCL. Sending multiple SendMatchOut_V1 messages for the same open position simply overwrites the previous quantity to be matched out.

2.1.16 SendPriceAvgHead_V1

Message Type - PH

The SendPriceAvgHead_V1 method is used together with the SendPriceAvgLine_V1 method to request that a number of trades be replaced with one trade with a price equal to the average of the trades being replaced.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_MsgSetID	Out	Yes	.
4	al_PriceAvgID	In	Yes	
5	al_EntID	In	Yes	
6	as_BuySell	In	Yes	

7	al_AvgPrice	In	Yes	
8	as_AvgPriceText	In	Yes	
9	as_RoundType	In	Yes	
10	al_Qty	In	No	The quantity is an absolute number.

Special Considerations

The messages comprising a Price Averaging batch (i.e. one header and two or more trade lines) are transmitted as message set. The message set identifier (al_MsgSetID) output by this method uniquely identifies the message set that must be supplied with all the lines in the batch. As_MsgStartEnd must be 'E' for the last trade line and 'M' for all other lines.

2.1.17 SendPriceAvgLine_V1

Message Type - PL

The SendPriceAvgLine_V1 method is used to identify trades that are to be included as part of a Price Averaging request. Two or more lines are batched together. Refer to SendPriceAvgHead_V1 for more details.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_MsgSetID	In	Yes	Refer to SendPriceAvgHead_V1
4	as_MsgStartEnd	In	Yes	Refer to SendPriceAvgHead_V1
5	al_TrlD	In	Yes	

2.1.18 SendPriceAvgUndo_V1

Message Type - PU

The SendPriceAvgUndo_V1 method is used to request the reversal of an earlier Price Averaging request.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_PriceAvgID	In	Yes	

2.1.19 SendReconLog_V1

Message Type - RL

The SendReconLog_V1 method is used to notify ASXCL of reconciliation discrepancies that have been identified by comparing data files transmitted by ASXCL with data stored at the participant site.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_LogType	In	Yes	
3	as_ReconcileID	In	Yes	
4	as_Logtext	In	Yes	

2.1.20 SendReservedCash_V1

Message Type - RC

The SendReservedCash_V1 method is used to notify ASXCL of reserved cash lodgements and withdrawal requests.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_AccID	In	Yes	
3	al_ActType	In	Yes	
4	al_Cur	In	Yes	
5	al_Amt	In	Yes	This is a cumulative amount.

2.1.21 SendRestartMbrProc_V1

Message Type - RM

The SendRestartMbrProc_V1 method is used to notify ASXCL that data files have been received and that the participant application is ready to commence processing.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	

Special Considerations

This method should be called after successful receipt and processing of end of day or Refresh file transmissions and following the receipt of a GetRefreshEnd_V1 or GetStartNewDayEnd_V1 message. It indicates to ASXCL that all subsequent messages received from the site should be processed.

2.1.22 SendStatusCheck_V1

Message Type - SC

The SendStatusCheck_V1 method is used to determine whether or not the participant site is able to communicate with ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_StatusCheckSeq	In	Yes	

Special Considerations

On receipt of this message, ASXCL automatically responds with a Status Check Response message. For more information see GetStatusCheckResponse_V1.

2.1.23 SendTakeUp_V1

Message Type - TU

The SendTakeUp_V1 method is used to notify ASXCL that a trade has been taken-up.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	No	
2	al_ExchID	In	No	
3	al_TrID	In	No	
4	as_AcceptFlag	In	No	
5	as_RejReason	In	No	

Special Considerations

Once a take-up message has been sent it cannot be undone. The participant to whom the trade has been given-up has the ability to return the trade via a give-up. ASXCL also has the ability to delete trades, which have been taken-up.

2.1.24 SendTRActTransferAccept_V1

Message Type – TY

The SendTrTransferAccept_V1 method is used to accept trading transfer activity received from another participant.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	No	
2	al_TransID	In	No	

3	al_ReqSeq	In	No
4	al_AccID	In	No
5	ac_ToMbrUnitFees	In	No
6	as_ToMbrReason	In	No

2.1.25 SendTRActTransferReject_V1

Message Type - TN

The SendTrActTransferReject_V1 method is used to decline the transfer of trading activity from another participant.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	No	
2	al_TransID	In	No	
3	al_ReqSeq	In	No	
4	as_ToMbrReason	In	No	

2.1.26 SendTRActTransferRequest_V1

Message Type - TF

The SendTransferMbrHead_V1 method is used together to request the transfer of trading activity from one Participant account to another Participant.

Trading Activity Transfers are transfers are used for “Day 2” corrections. Trading Activity Transfers allow fees and commissions to be transferred and facilitate the retention of the date of trade and price.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	No	
2	al_TransID	In	No	
3	al_ReqSeq	In	No	
4	al_FromAccID	In	No	
5	al_ToMbrID	In	No	
6	as_FailedGiveUp	In	No	
7	al_TransferQty	In	No	
8	ac_TransferComm	In	No	
9	as_FromMbrReason	In	No	
10	as_MbrInfo	In	No	

2.1.27 SendTransferHead_V1

Message Type - TH

The SendTransferHead_V1 method is used together with the SendTransferLine_V1 method to request the transfer of one or more open positions from one account to another account within the same participant.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_MsgSetID	Out	Yes	
3	as_Origin	In	Yes	
4	al_TransID	In	Yes	
5	al_PositionLines	In	Yes	
6	al_FromAccID	In	Yes	
7	al_ToAccID	In	Yes	
8	as_OpenClose	In	Yes	
9	as_Comment	In	Yes	
10	as_MbrInfo	In	No	
11	al_SupportLines	In	Yes	

Special Considerations

Account to account transfers must be approved by ASXCL. Refer to GetTransferAcceptReject_V1 for more information.

The messages comprising a transfers batch (i.e. one header message, one or more transfer lines and one or more support info lines) are transmitted as a message set. The message set identifier (al_MsgSetId) output by this method uniquely identifies the message set and must be supplied with all the lines in the batch. As_MsgStartEnd must be 'E' for the last transfer line and 'M' for all the other lines.

2.1.28 SendTransferLine_V1

Message Type - TL

The SendTransferLine_V1 method is used to identify the position to be transferred from one account to another account within the same participant. One or more transfer lines are batched for transfers between the same accounts. Refer to *SendTransferHead_V1* method for more information.

Sequence	Argument	In/Out	Mandatory	Notes

1	as_UserID	In	Yes	
2	al_MsgSetID	In	Yes	Refer to SendTransferHead_V1
3	as_MsgStartEnd	In	Yes	Refer to SendTransferHead_V1
4	al_LineNum	In	Yes	Starting from one, incremented by one for each line.
5	al_EntID	In	Yes	
6	as_BuySell	In	Yes	
7	al_Qty	In	Yes	The quantity is an absolute number.

2.1.29 SendTransferMbrAccept_V1

Message Type - TA

The SendTransferMbrAccept_V1 method is used to accept all the positions in a transfer batch received from another participant. Refer to

*GetCHTransferMbrHead*_V1*/GetCHTransferMbrLine_V1* for more information. The method is also used to inform ASXCL of the account into which the positions are to be transferred.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_Origin	In	Yes	
3	al_TransID	In	Yes	
4	al_AccID	In	Yes	Identifies the receiving account.
5	as_OpenClose	In	No	Not Used
6	as_Comment	In	Yes	

Special Considerations

A transfer between participants can be accepted before or after it has been approved by ASXCL. The transfer is not complete until it has been accepted by the transferee and approved by ASXCL.

2.1.30 SendTransferMbrHead_V1

Message Type - MH

The SendTransferMbrHead_V1 method is used together with the SendTransferLine_V1 method to request the transfer of one or more open positions from one account to another participant.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	No	
2	al_MsgSetID	Out	No	
3	as_Origin	In	No	
4	al_TransID	In	No	
5	al_PositionLines	In	No	
6	al_FromAccID	In	No	
7	al_ToMbrID	In	No	
8	as_ToAccID	In	No	
9	ac_Amt	In	No	
10	as_Cur	In	No	
11	as_Ledger	In	No	
12	as_OpenClose	In	No	Not used.
13	as_Comment	In	No	
14	as_MbrInfo	In	Yes	
15	al_SupportLines	In	No	

Special Considerations

Account to account transfers must be accepted by the transferee and approved by ASXCL. Refer to *GetCHTransferAccept_V1/ GetCHTransferReject_V1* for more information.

2.1.31 SendTransferMbrLine_V1

Message Type - ML

The SendTransferMbrLine_V1 method is used to identify the position to be transferred to another participant. One or more transfer lines are batched for transfers between the same parties - refer to *SendTransferMbrHead_V1*method for more information.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_MsgSetID	In	Yes	Refer to SendTransferHead_V1
3	as_MsgStartEnd	In	Yes	Refer to SendTransferHead_V1
4	al_LineNum	In	Yes	
5	al_EntID	In	Yes	
6	as_BuySell	In	Yes	

7	al_Qty	In	Yes	The quantity is an absolute number.

2.1.32 SendTransferMbrReject_V1

Message Type - TJ

The SendTransferMbrReject_V1 method is used to decline the transfer of a batch of positions from another participant.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	as_Origin	In	Yes	
3	al_TransID	In	Yes	
4	as_Comment	In	Yes	Reason for rejection.

Special Considerations

The transferee may reject a transfer if it has been approved by ASXCL, but not after it has been accepted (by the transferee).

2.1.33 SendTransferSupportLine_V1

Message Type - TS

The SendTransferSupportLine_V1 method is used to send information to support the transfer request. The supporting information required is defined by ASXCL. This method is used for both participant-to-participant transfers and Account-to-Account transfers.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_MsgSetID	In	Yes	Refer to SendTransferhead_V1.
3	as_MsgStartEnd	In	Yes	Refer to SendTransferhead_V1.
4	as_SupportInfo	In	Yes	Reason for transfer.

2.1.34 SendUndoAlloc_V1

Message Type - UA

The SendUndoAlloc_V1 method is used to notify ASXCL that a previously transmitted allocation is to be reversed.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_TrID	In	Yes	
4	al_AllocSeq	In	Yes	
Special Considerations				
The allocation sequence associated with an allocation that has been undone should not be reused.				

2.1.35 SendUndoMatchOutRequest_V1

Message Type - UM

The SendUndoMatchOutRequest_V1 method is used to notify ASXCL of a request for a matchout reversal.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_UserID	In	Yes	
2	al_ExchID	In	Yes	
3	al_AccID	In	Yes	
4	al_EntID	In	Yes	
5	adt_MatchOutDate	In	Yes	
6	al_QtyMatchedOut	In	Yes	
7	al_QtyReopen	In	Yes	
8	as_Reason	In	Yes	
9	al_UndoID	In	No	

Special Considerations
This method is used to request a reversal of matchouts made prior to the current business day. To undo matchouts made on the current day, see SendMatchOut_V1. When ASXCL undoes the matchout, the message GetPosAdj_V1 is sent.

2.2. Messages Received from ASXCL

Messages received from ASXCL contain methods of the RecMsg ActiveX control and their arguments.

Note that where an argument is described as being optional it indicates that the **value** is optional (i.e. depending on the data type values of 0 or "" are permissible), however the argument must always be provided. Arguments that are described as "In" must be supplied when activating the method. Arguments that are described as "Out" are those that are returned by the method.

Note:

The use of single quotation marks and tab characters are not permitted in any arguments.

For each method a two character Message Type is shown. This is the code associated with the particular message, i.e. the value of ls_MsgType when using the GetNextMsg method. The Message Type is largely irrelevant for messages sent to ASXCL since it is automatically determined by the SendMsg ActiveX control, depending upon the method invoked.

2.2.1 GetBCast_V1

Message Type – BC and MA (mail)

The GetBCast_V1 method is used obtain details of a message broadcast sent to the participant site by ASXCL. Examples of message broadcasts include:

- ASXCL correspondence
- Notification of rejections
- Intra-day margin calls
- Notification of pending cut-offs; etc.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_BCastID	Out	Yes	
3	adt_BCastDate	Out	Yes	
4	as_BCastType	Out	Yes	
5	as_BCastTitle	Out	Yes	
6	as_BCastText	Out	Yes	
7	as_AttachName	Out	No	
8	as_AttachType	Out	No	Will be present when as_AttachName contains a value.
9	al_MsgSeq	Out	No	If the message broadcast relates to an error detected by ASXCL then this argument contains the sequence number of the message that was rejected by ASXCL.
10	as_MsgTag	Out	No	

Special Considerations

Should a message broadcast have an attachment, this message would have been preceded by a file transfer message containing the information necessary to perform the transfer. See GetFileTransfer_V1 for more information.

2.2.2 GetCHTransferSupportLine_V1

Message Type – TS

The details to be received only.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	As_SupportInfo1	Out	Yes	
3	As_SupportInfo2	Out	Yes	
4	As_SupportInfo3	Out	Yes	

Special Considerations

The number of lines in the batch header should include the support line. If MCM Transfer results in an error of 51031, this indicates that the number of lines does not match.

2.2.3 GetUndoMatchOutRequestRej_V1

Message Type – UJ

The GetUndoMatchOutRequestRej_V1 method is used to get the details of any undo match-out requests that have been rejected by ASXCL. Refer to SendUndoMatchOutRequest_V1 for more information.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_CHReason	In	Yes	
3	al_UndoIDMethod Arguments	In	Yes	

2.2.4 GetMarketMarkerObligations_V1

Message Type – MK

The GetMarketMakerObligations_V1 method is used to obtain details of changes to the market maker obligation parameters during the day. It should be noted that market maker obligations are distributed nightly by the file transfer mechanism.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	As_AmendmentType	In	Yes	
3	Al_MbrID	In	Yes	
4	As_DerivProd	In	Yes	

2.2.5 GetBulkTransferHead_V1

Message Type – BH

The GetBulkTransferHead_V1 method is used together with the GetBulkTransferLine_V1 method to obtain details of positions transferred by ASXCL on the participant behalf. The details may refer to positions transferred in or out.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_Origin	Out	Yes	
3	al_TransID	Out	Yes	
4	al_BatchLines	Out	Yes	
5	al_OtherMbrID	Out	Yes	
6	al_AccID	Out	Yes	
7	ac_Cash	Out	No	
8	as_Cur	Out	No	
9	as_Comment	Out	Yes	
10	as_InOut	Out	Yes	
11	al_ToAccID	Out	Yes	

Special Considerations

No acceptance is required for bulk transfers.

2.2.6 GetBulkTransferLine_V1

Message Type – BL

The GetBulkTransferLine_V1 method is used to obtain details of each position line transferred. Refer to GetBulkTransferHead_V1 method for more information.

Sequence	Argument	In/Out	Mandatory	Notes

1	as_MsgText	In	Yes	
2	al_LineNum	Out	Yes	
3	al_EntID	Out	Yes	
4	as_BuySell	Out	Yes	
5	al_Qty	Out	Yes	The quantity is an absolute number.

Special Considerations

No acceptance is required for bulk transfers.

2.2.7 GetBuyTenderDetails_V1

Message Type – DD

The GetBuyTenderDetails_V1 method is used obtain details relating to tenders assigned to the participant by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_AccID	Out	Yes	
3	al_EntID	Out	Yes	
4	as_PhysRef	Out	Yes	
5	ac_Amt	Out	Yes	
6	adt_TenderDate	Out	Yes	

2.2.8 GetCHAcc_V1

Message Type – AH

The GetCHAcc_V1 method is used to obtain details of account changes made by ASXCL. These changes relate to fields that only ASXCL are allowed to change.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	as_AmendmentType	Out	No	Always ‘E’.
3	al_AccID	Out	No	
4	as_CoverGrp	Out	No	
5	as_AccType	Out	No	

6	al_AssocMbrID	Out	No	Zero when not applicable.
7	al_EqtClearID	Out	No	
8	as_DefRT	Out	No	
9	Adt_DateCreated	Out	No	
10	as_PaperlessColl	Out	No	

2.2.9 GetCHAccName_V1

Message Type – AN

The GetCHAccName_V1 method is used to ascertain whether an account name change has been accepted or rejected by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_AccID	Out	Yes	
3	as_AccName	Out	Yes	The account name as recorded by ASXCL. This is the new name if the change is accepted or the old name if the change was rejected.
4	as_AcceptReject	Out	Yes	

Special Considerations

as_AccName is always present and reflects the account name currently recorded by ASXCL for this account. Refer to SendAcc_V1 for more information.

2.2.10 GetCHAlloc_V1

Message Type – AA

The GetCHAlloc_V1 method is used to obtain details of an automatic allocation performed by ASXCL on receipt of a trade from the exchange. It contains the account code to which the trade is to be allocated.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_TrID	Out	Yes	
3	al_AllocSeq	Out	Yes	
4	al_Qty	Out	Yes	The quantity is an absolute number.
5	al_AccID	Out	Yes	
6	as_OpenClose	Out	No	Reserved for future use.
7	ac_Comm	Out	No	
8	ac_CommBasisVal	Out	No	
9	as_CommBasis	Out	No	

Special Considerations

This message is immediately preceded by a trade message to which this allocation relates. See GetTrade_V1 for more information.

2.2.11 GetCHGiveUp_V1

Message Type – CG

The GetCHGiveUp_V1 method is used to obtain details of an automatic give-up performed by ASXCL on receipt of a trade from the exchange. This contains the participant code that the trade is to be given-up.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_TrID	Out	Yes	
3	al_AllocSeq	Out	Yes	
4	al_Qty	Out	Yes	The quantity is an absolute number.
5	al_MbrFor	Out	Yes	
6	ac_Comm	Out	Yes	

7	ac_CommBasisVal	Out	No
8	as_CommBasis	Out	No

Special Considerations

This message is immediately preceded by a trade message to which the allocation relates. See GetTrade_V1 for more information.

2.2.12 GetCHStatusCheck_V1

Message Type – ST

The GetCHStatus_V1 method is used to obtain details of a status check issued by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_StatusCheckSeq	Out	Yes	

Special Considerations

On receipt of this message the SendCHStatusCheckResponse_V1 method should be called to notify the exchange that the status check message has been received.

2.2.13 GetCHTrActTransferAccept_V1

Message Type – CY

The GetCHTrActTransferAccept_V1 method is used to obtain details relating to the acceptance of a trade activity transfer by the transferee or ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	al_TransID	Out	No	
3	al_ReqSeq	Out	No	
4	as_Reason	Out	No	
5	ac_UnitFees	Out	No	
6	ac_UnitPremium	Out	No	
7	ac_UnitMTM	Out	No	
8	ac_UnitContVal	Out	No	
9	ac_UnitMktVal	Out	No	
10	as_AcceptedBy	Out	No	
11	al_NewTransID	Out	No	

2.2.14 GetCHTrActTransferRequest_V1

Message Type – CF

The GetCHTrActTransferRequest_V1 method is used to obtain details of the trading activity transferred.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	al_TransID	Out	No	
3	al_ReqSeq	Out	No	
4	al_FromMbrID	Out	No	
5	al_EntID	Out	No	
6	as_BuySell	Out	No	
7	as_TrDerivProd	Out	No	
8	as_TrEntDesc	Out	No	
9	adt_TrDate	Out	No	
10	al_TrPrice	Out	No	
11	as_TrPriceText	Out	No	
12	ac_CHToMbrUnitFees	Out	No	
13	as_CapAdj	Out	No	
14	al_CapAdjQtyFactor	Out	No	
15	as_FailedGiveUp	Out	No	
16	al_TransferQty	Out	No	
17	ac_TransferComm	Out	No	
18	as_CHStatus	Out	No	
19	as_CHReason	Out	No	
20	as_FromMbrReason	Out	No	

2.2.15 GetCHTrActTransferReject_V1

Message Type – CN

The GetCHTrActTransferReject_V1 method is used to obtain details relating to the rejection of a trade activity transfer by the transferee or ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	

2	al_TransID	Out	No
3	al_ReqSeq	Out	No
4	as_Reason	Out	No
5	as_RejectedBy	Out	No

2.2.16 GetCHTransferAccept_V1

Message Type – CA

The GetCHTransferAccept_V1 method is used to obtain details relating to the acceptance of a batch of transfers by the transferee or ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_Origin	Out	Yes	
3	al_TransID	Out	Yes	
4	as_AcceptedBy	Out	Yes	
5	as_Comment	Out	Yes	

2.2.17 GetCHTransferMbrHead_V1

Message Type – CH

The GetCHTransferMbrHead_V1 method is used together with the GetCHTransferMbrLine_V1 method to obtain details of a transfer of positions requested by another participant. The details refer to positions to be received only.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	as_Origin	Out	No	
3	al_TransID	Out	No	
4	al_BatchLines	Out	No	
5	al_FromMbrID	Out	No	
6	as_ToAcc	Out	Yes	
7	ac_Amt	Out	No	
8	as_Cur	Out	No	
9	as_Ledger	Out	No	
10	as_Comment	Out	No	

Special Considerations
Transfers received from another participant must be accepted or rejected. Refer to SendTransferMbrAccept_V1/SendTransferMbrReject_V1 for more information.

2.2.18 GetCHTransferMbrLine_V1

Message Type – CL

The GetCHTransferMbrLine_V1 method is used to obtain details of each position line transferred. Refer to

GetCHTransferMbrHead_V1 for more information.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_LineNum	Out	Yes	
3	al_EntID	Out	Yes	
4	as_BuySell	Out	Yes	
5	al_Qty	Out	Yes	The quantity is an absolute number.

2.2.19 GetCHTransferReject_V1

Message Type – CJ

The GetTransferReject_V1 method is used to obtain details relating to the rejection of a batch of transfers by the transferee or ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_Origin	Out	Yes	
3	al_TransID	Out	Yes	
4	as_RejectedBy	Out	Yes	
5	as_Comment	Out	Yes	

2.2.20 GetCollateralActivity_V1

Message Type – CM

The GetCollateralActivity_V1 method is used to obtain details of changes to collateral holdings at ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_SegType	In	Yes	

3	as_GroupLevel	In	Yes	
4	al_AccID	In	Yes	Zero if as_GroupLevel is Y. Empty if as_GroupLevel is N.
5	as_CoverGrp	In	Yes	
6	as_LodgeID	In	Yes	
7	as_CollActType	In	Yes	
8	as_CollType	In	Yes	
9	as_IssuerCode	In	Yes	
10	adt_LodgeDate	In	Yes	
11	as_CollDetail	In	Yes	
12	as_UnitCode	In	Yes	
13	al_Units	In	Yes	
14	as_Holder	In	Yes	
15	As_HIN	In	Yes	
16	As_SpecificCover	In	Yes	
17	Adt_ExpiryDate	In	Yes	
18	As_Cur	In	Yes	

2.2.21 GetDerivProd_V2

Message Type – DP

The GetDerivProd_V2 method is used to obtain details of changes to derivative product parameters that have taken place during the day.

Note:

Derivative product parameters are distributed nightly by the file transfer mechanism. Changes to derivative product details during the course of the day are likely to be infrequent.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_AmendmentType	Out	Yes	
3	as_DerivProd	Out	Yes	
4	as_DerivProdStat	Out	Yes	
5	as_DerivProdType	Out	Yes	
6	as_DerivProdDesc	Out	Yes	
7	al_SettPrice	Out	Yes	

8	as_SettPriceText	Out	Yes
9	as_Basket	Out	Yes
10	as_ManAutoExer	Out	Yes
11	As_AllowExer	Out	Yes

2.2.22 GetDepositoryActivity_V1

Message Type – HA

The GetDepositoryActivity_V1 method is used to obtain details of changes to commodity (e.g. grain and wool) holdings at ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	al_AccID	Out	No	
3	as_DepositID	Out	No	
4	al_ActID	Out	No	
5	as_ActDesc	Out	No	
6	al_Units	Out	No	
7	as_DelProd	Out	No	
8	as_HeldAt	Out	No	
9	as_CPRef	Out	No	
10	as_MoreDetails	Out	No	
11	adt_ActTime	Out	No	

2.2.23 GetEquityClearer_V1

Message Type – EQ

The GetEquityClearer_V1 method is used to obtain details of changes to ASX Equity Clearer details that have taken place during the day.

Note:

Equity Clearer parameters are distributed nightly by the file transfer mechanism. Changes to Equity Clearer details during the course of the day are likely to be infrequent.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_AmendmentType	Out	Yes	
3	al_EqtClearID	Out	Yes	
4	as_EqtClear	Out	Yes	

5	as_EqtClearName	Out	Yes

2.2.24 GetFacility_V1

Message Type – FC

The GetFacility_V1 method is used to obtain details of which transactions are currently accepted by ASXCL. Facility messages are sent by ASXCL when a transaction type reaches its cut-off and transactions of this type are no longer accepted by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_GiveUp	Out	Yes	
3	as_TakeUp	Out	Yes	
4	as_TransferEntry	Out	Yes	
5	as_TransferAccept	Out	Yes	
6	as_Exercise	Out	Yes	
7	as_ShutDown	Out	Yes	
8	as_CashWithdrawals	Out	Yes	

Special Considerations

Should the participant site continue to send transactions that are no longer accepted by ASXCL, then the transactions are rejected.

2.2.25 GetFileTransfer_V1

Message Type – FT

The GetFileTransfer_V1 method is used to obtain file data transmitted from ASX. On receipt of this message the participant can access the file name that is stored in the as_URLName argument. In addition, the size of the file transferred can be compared to the al_FileSize argument to ensure integrity. The file is located in the standard location determined by the Windows registry setting on the MCM Server. The standard location is: *C:\MCM_Data\FileTransfers*. The location is found in the registry setting *HKEY_LOCAL_MACHINE\SOFTWARE\Palion Pty Ltd\MCM\1.0\FileTransferDir*.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	as_Description	Out	No	
3	as_FileType	Out	No	

4	as_FileName	Out	No
5	al_FileSize	Out	No

Special Considerations

For security measures, the location of files is not predictable.

2.2.26 GetGUAdvice_V1

Message Type – GA

The GetGUAdvice_V1 method is used to obtain details relating to whether a give-up has been accepted or rejected by the other party, or in response to a successful undo give-up request.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	al_TrID	Out	No	
3	al_AllocSeq	Out	No	
4	as_AcceptFlag	Out	No	
5	as_RejReason	Out	No	

Special Considerations

The allocation sequence associated with a rejected give-up allocation should not be reused.

ASXCL also issues a BC message type.

2.2.27 GetGUDeletion_V1

Message Type – GD

The GetGUDeletion_V1 method is used obtain details of a trade that has been taken-up but has been subsequently deleted by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_TrID	Out	Yes	

Special Considerations

ASXCL will issue a BC message.

2.2.28 GetMbr_V1

Message Type – MB

The GetMbr_V1 method is used to obtain details of changes to participant parameters that have taken place during the day.

Note:

Participant parameters are distributed nightly by the file transfer mechanism. Changes to participant details during the course of the day are likely to be infrequent.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_AmendmentType	Out	Yes	
3	al_MbrID	Out	Yes	
4	as_Mbr	Out	Yes	
5	as_MbrClearType	Out	Yes	
6	as_MbrName	Out	Yes	
7	as_MbrShortName	Out	Yes	

2.2.29 GetMembership_V1

Message Type – MM

The GetMembership_V1 method is used to obtain details of changes to participant parameters that have taken place during the day.

Note:

It should be noted that participant parameters are distributed nightly by the file transfer mechanism. Changes to participant details during the course of the day are likely to be infrequent.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_AmendmentType	Out	Yes	
3	al_ExchID	Out	Yes	
4	al_MbrID	Out	Yes	
5	al_DefAccID	Out	No	Will only contain a value if the message relates to the participant receiving the message.
6	al_DefHouseAccID	Out	No	Will only contain a value if the message relates to the participant receiving the message.

7	al_DefClientAccID	Out	No	Will only contain a value if the message relates to the participant receiving the message.
8	al_ClearMbrID	Out	No	Will only contain a value if the message relates to the participant receiving the message.
9	as_Active	Out	Yes	

2.2.30 GetPosAdj_V1

Message Type – PA

The GetPosAdj_V1 method is used to obtain details of a position adjustment transaction performed by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_AccID	Out	Yes	
3	al_EntID	Out	Yes	
4	as_BuySell	Out	Yes	
5	al_Qty	Out	Yes	al_Qty can be a positive or a negative value.
6	al_UndoID	Out	No	
7	as_CHReason	In	No	

Special Considerations

This method is used by ASXCL for position adjustments and undo matchouts. al_UndoID = 0 signifies no (empty/null/etc.) UndoID, an empty string for as_CHReason signifies no reason. If UndoID <> 0, the UnodMatchOutReq table is updated.

2.2.31 GetPriceAvgAct_V1

Message Type – PD

The GetPriceAvgAct_V1 method is used to obtain details of a Price Averaging request that has been actioned by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_PriceAvgID	Out	Yes	
3	al_NewTrID	Out	Yes	
4	ac_RoundDiff	Out	Yes	

2.2.32 GetRefreshEnd_V1

Message Type – RE

The GetRefreshEnd_V1 method is used to obtain details of the last message in a refresh message set.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_RefreshType	Out	Yes	

Special Considerations

A refresh message set consists of a:

Start (see GetRefreshStart_V1)

Several file transfers (see GetFileTransfer_V1)

An end (this message).

2.2.33 GetRefreshStart_V1

Message Type – RS

The GetRefreshStart_V1 method is used to obtain details relating to the first message in a refresh message set.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_RefreshType	Out	Yes	

Special Considerations

A refresh message set consists of a:

Start (this message)

Several file transfers (see GetFileTransfer_V1)

An end (see GetRefreshEnd_V1).

2.2.34 GetSoftwareUpgrade_V1

Message Type – SU

The GetSoftwareUpgrade_V1 method is used to obtain file data transmitted from ASX. On receipt of this message the participant can access the file whose name is stored in the as_URLName argument. In addition, the size of the file transferred can be compared to the al_FileSize argument to ensure integrity.

The file is located in the standard location determined by the Windows registry setting on the MCM Server.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_SoftVer	Out	Yes	
3	as_FileName	Out	Yes	
4	as_FileType	Out	Yes	New File Type: X for XML files.
5	as_URLName	Out	Yes	
6	al_FileSize	Out	No	

Special Considerations

As a security measure, the location of files is not predictable.

2.2.35 GetStartNewDayEnd_V1

Message Type – SE

The GetStartNewDayEnd_V1 method is used to obtain details of the last message in a start new day message set.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	adt_BusDate	Out	Yes	

Special Considerations

A start new day message set consists of:

A start (see GetStartNewDayStart_V1)

Several file transfers (see GetFileTransfer_V1)

An end (this message).

2.2.36 GetStartNewDayStart_V1

Message Type – SS

The GetStartNewDayStart_V1 method is used to obtain details relating to the first message in a start new day message set.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	adt_BusDate	Out	Yes	

Special Considerations

A start new day message set consists of:

A start (this message)

Several file transfers (see GetFileTransfer_V1)

An end (see GetStartNewDayEnd_V1).

2.2.37 GetStatusCheckResponse_V1

Message Type – SR

The GetStatusCheckResponse_V1 method is used to obtain details relating to the response from a status check message (see *SendStatusCheck_V1)* issued by the participant site. The receipt of this message indicates that communications with ASXCL are functioning correctly.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_StatusCheckSeq	Out	Yes	

Special Considerations

The al_StatusCheckSeq contains the sequence number that was sent to ASXCL using the SendStatusCheck_V1 message.

2.2.38 GetTenderDetails_V1

Message Type – DD

The GetTenderDetails_V1 method is used obtain tender details.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	al_AccID	Out	No	
3	al_EntID	Out	No	
4	al_OtherMbrID	Out	No	
5	as_PhysRef	Out	No	

6	ac_Amt	Out	No
7	ac_GSTVal	Out	No
8	adt_TenderDate	Out	No
9	al_Qty	Out	No
10	as_BuySell	Out	No
11	al_TenderStyle	Out	No
12	as_Final	Out	No
13	as_TenderDetails	Out	No
14	al_DelDays	Out	No

2.2.39 GetTenderAdvice_V1

Message Type – DA

The GetTenderDetails_V1 method is used obtain tender details.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	al_AccID	Out	Yes	
3	al_EntID	Out	Yes	
4	al_Qty	Out	Yes	The quantity is a relative number.
5	as_BuySell	Out	Yes	

2.2.40 GetTrade_V1

Message Type – TR

The GetTrade_V1 method is used obtain details of a trade transmitted by ASXCL. The trade details may relate to a standard trade, a contra trade or a take-up trade.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	No	
2	al_TrID	Out	No	
3	al_EntID	Out	No	
4	as_MktMaker	Out	No	
5	al_ExecMbrID	Out	No	
6	as_Trader	Out	No	
7	al_OtherMbr	Out	No	
8	as_ExchRef	Out	No	
9	as_BuySell	Out	No	

10	adt_TrDate	Out	No	
11	al_TrPrice	Out	No	
12	as_TrPriceText	Out	No	
13	al_Qty	Out	No	This quantity is an absolute number.
14	as_Origin	Out	No	
15	ac_Comm	Out	Yes	
16	as_Ref	Out	Yes	
17	as_Acc	Out	Yes	
18	as_CompType	Out	Yes	
19	al_CompParts	Out	No	
20	as_TraderType	Out	Yes	
21	adt_TrTime	Out	No	
22	adt_RecTime	Out	No	
23	as_Contra	Out	No	
24	al_OrigTRID	Out	Yes	
25	al_AllocSeq	Out	Yes	
26	al_PriceAvgID	Out	Yes	
27	ac_UnitContVal	Out	No	
28	as_ConditionCodes	Out	Yes	
29	as_EFP	Out	Yes	
30	ac_CommBasisVal	Out	Yes	
31	as_CommBasis	Out	Yes	
32	as_TrOrderNo	Out	Yes	

2.2.41 GetTradeDeletion_V1

Message Type – TD

The GetTradeDeletion_V1 method is used to obtain details of trades that have been deleted by ASXCL.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In		
2	al_TrID	Out		

Special Considerations
If part of the traded quantity has been allocated, ASXCL issues a Broadcast Message (message type = BC) requesting the participant to delete the allocation.

2.2.42 GetTradedEntity_V2

Message Type – TE

The GetTradedEntity_V2 method is used to obtain details of changes to traded entity parameters that have taken place during the day.

Note:

Traded entity parameters are distributed nightly by the file transfer mechanism. Changes to traded entity details during the course of the day are likely to be infrequent.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_AmendmentType	Out	Yes	
3	al_EntID	Out	Yes	
4	as_DerivProd	Out	Yes	
5	as_OptType	Out	Yes	
6	adt_DelMonth	Out	Yes	
7	adt_ExpDate	Out	Yes	
8	al_Strike	Out	Yes	
9	as_StrikeText	Out	Yes	
10	al_Version	Out	Yes	
11	as_VersionText	Out	Yes	
12	as_SecCode	Out	Yes	
13	ac_UnitsPerLot	Out	Yes	
14	adt_CashSettDate	Out	Yes	
15	as_EntityShortCut	Out	Yes	
16	as_ExerStyle	Out	Yes	
17	ac_Multiplier	Out	Yes	
18	as_TradingCeased	Out	Yes	
19	as_Spot	Out	Yes	
20	adt_DelStartDate	Out	Yes	
21	as_DelMonthText	Out	Yes	
22	as_IsOTC	Out	Yes	

2.2.43 GetTransferConf_V1

Message Type – TC

The GetTransferConf_V1 method is used to obtain details relating to transfers, which have been accepted/approved by all parties.

Sequence	Argument	In/Out	Mandatory	Notes
1	as_MsgText	In	Yes	
2	as_Direction	Out	Yes	
3	al_LineNum	Out	Yes	
4	al_AccID	Out	Yes	
5	al_EntID	Out	Yes	
6	as_BuySell	Out	Yes	
7	al_Qty	Out	Yes	The quantity is an absolute number.

Special Considerations

The participant may choose to monitor the progress of transfers via the array of other transfer messages. However if the participant system is only interested in the final outcome of the transfer, then this message process is used.

2.2.44 Other Messages

Other messages that may be issued by ASXCL consist solely of a message type. Consequently they do not have an associated method. These message types are described below.

Message Type	Comment
CP	Collateral Prices This message confirms that a recent file transfer of Collateral Prices has been initiated.
IC	Intra-day Capital Adjustment This message confirms that a recent file transfer of capital adjustments has been initiated.
IR	Intra-day Reference Price This message confirms that a recent file transfer of reference prices has been initiated.
IT	Intra-day Theoretical Prices This message confirms that a recent file transfer of theoretical prices has been initiated.
RP	Restart Processing This message indicates that processing of messages on the standard message queue should recommence.
UR	Update Reference Prices This message advises that Member Reference Prices should be replaced with the Reference Prices issued by ASXCL

3. Method Argument Definitions

The table below defines the arguments that are passed to the SendMsg and RecMsg ActiveX controls. The definitions are of a general nature. Refer to notes accompanying the message definition for any overriding variations.

Name	Data Type	Length	Description
ac_Amt	Currency	8	The amount paid/received as a result of the tender. OR The amount of reserved cash to lodge/withdraw.
ac_Cash	Currency	8	The cash value to accompany a position transfer. The value cannot be negative.
ac_Comm	Currency	8	The commission amount that is payable for a trade that is taken-up and receivable for a trade which is given-up. The value cannot be negative.
ac_CommBasisVal	Currency	8	The rate per lot or the percentage of deal value or a total amount of commission charged in relation to a

			give-up or an allocation to a consolidated account.
ac_CHToMbrUnitFees	Currency	8	The fee per contract.
ac_GSTVal	Currency	8	The GST amount related to a transaction, for example, a tender.
ac_GUComm	Currency	8	The default give-up commission rate per lot/percentage of deal value which would usually be charged in relation to a give-up to a specific participant for a particular derivative product. The commission value cannot be negative.
ac_Multiplier	Currency	8	The multiplier used for contract value calculations.
ac_RoundDiff	Currency	8	The rounding difference resulting from the calculation of a Price Average request.
ac_TUComm	Currency	8	The default take-up commission rate per lot/percentage of deal value which would usually be payable in relation to a trade taken-up from a specific participant for a particular derivative product. The commission value cannot be negative.
ac_UnitContVal	Currency	8	The value of one contract calculated at the traded price.
ac_UnitsPerLot	Currency	8	The number of shares per contract.
ac_UnitContVal	Currency	8	The traded value of one contract.
ac_UnitFees	Currency	8	The fees per contract.
ac_UnitMktVal	Currency	8	The market value of one contract.
ac_UnitMTM	Currency	8	The settlement to market amount per contract.
ac_UnitPremium	Currency	8	The premium per contract.
al_TrPrice	Double	8	The price at which a trade was done. Note that this price is unformatted and does not contain any decimal places.
adt_ActTime	Date	8	The date and time of an activity.
adt_BcastDate	Date	8	The date and time at which the message broadcast was sent by ASXCL.
adt_BusDate	Date	8	The current business date.

adt_CashSettDate	Date	8	The date on which cash settlement will take place.
adt_DelMonth	Date	8	The delivery date associated with a traded entity. Note that delivery dates are formatted as MMMYY, e.g. SEP14 and therefore the day component of these dates are always pre-set to 01.
adt_DelStartDate	Date	8	The date that the first deliverable future can be tendered.
adt_ExpDate	Date	8	The date on which a traded entity is due to expire.
adt_LodgeDate	Date	8	The date collateral was lodged at ASXCL.
adt_MatchoutDate	Date	8	The date of the original matchout.
adt_RecTime	Date	8	The date and time at which a trade was received by ASXCL from an Exchange.
adt_TenderDate	Date	8	The date on which the notice of tender was accepted by ASXCL.
adt_TrDate	Date	8	The date on which a trade was executed.
adt_TrTime	Date	8	The date and time at which a trade was completed. The relevant exchange supplies this information to ASXCL.
adt_DateCreated	Date	8	The date of creation of a record.
as_HIN	String	10	The Holder Identification Number.
adt_ExpiryDate	Date	8	The date a collateral lodgement is due to expire.
as_Ledger	String	12	The code uniquely identifying a ledger.
as_CHReason	String	250	Information supplied by ASXCL to explain its actions.
al_UndoID	Long	4	The code uniquely identifying an ‘Undo Matchout’ request.
al_AccID	Long	4	A unique number assigned to each Account as it is added. The numbering must start at two and be incremented by one.
al_ActID	Long	4	A unique number assigned to a particular transaction or activity.

al_AllocSeq	Long	4	A unique number assigned to each allocation/give-up associated with a trade. The first allocation/give-up for each trade should be assigned the sequence number of 1. Subsequent allocations/give-ups should be one greater than the last allocation/give-up for the same trade.
al_AssocMbrID	Long	4	The Member ID of the Registered Independent Options Trader (RIOT) or the corporate used for RIOT and consolidated accounts only.
al_AvgPrice	Long	4	The calculated average price associated with a Price Average request.
al_CapAdjQtyFactor	Long	4	The multiplier that has been used to increase the number of open contracts due to corporate actions.
al_BatchLines	Long	4	The number of transfer lines contained within a transfer batch.
al_BCastID	Long	4	A unique number used to identify each Broadcast message.
al_ClearMbrID	Long	4	A unique number used to identify each participant.
al_CompParts	Long	4	The number of legs in a composite trade, or one for standard trades.
al_DefAccID	Long	4	The al_AccID of the Default Account.
al_DefHouseAccID	Long	4	The al_AccID of the House omnibus Account.
al_DefClientAccID	Long	4	The al_AccID of the Client omnibus Account.
al_DelDays	Long	4	The number of days before delivery is completed.
al_EntID	Long	4	A unique number used to identify each Traded Entity.
al_EqtClearID	Long	4	A unique number used to identify the equity clearer.
al_ExchID	Long	4	A unique number that identifies each Exchange. Exchange identifiers used by ASX are as follows: 1 = ASX - ETO 2 = ASXF 3 = n/a.

al_ExecMbrID	Long	4	A unique number that identifies the participant, Registered Trader (RT), RIOT or Non-Clearing participant who executed the trade.
al_FileSize	Long	4	The size of a file in bytes. This value is used to determine whether files transferred from ASXCL are the expected size.
al_FromAccID	Long	4	The al_AccID of an account that has contracts that are the subject of a transfer.
al_FromMbrID	Long	4	A unique number that identifies a participant who is transferring a position.
al_GrpID	Long	4	A unique number assigned to each Account Group as it is added. The numbering must start at one and be incremented by one.
al_LineNum	Long	4	Identifies each transfer line within a batch of transfers. Numbering should start at one and then incremented by one for each line of position being transferred.
al_MbrFor	Long	4	A unique number that identifies a participant to whom all or part of a trade is being given-up to.
al_MbrID	Long	4	A unique number that identifies a clearing participant, non-clearing participant, RT or RIOT.
al_MsgSeq	Long	4	A number that uniquely identifies a message within a message queue.
al_MsgSetID	Long	4	Contains the al_MsgSeq number of the first message within a message set. All messages within a message set have the same message set ID.
al_MsgVersion	Long	4	The version of the message. When changes are made to the format of a message a new version of the message is introduced.
al_NewTransID	Long	4	A unique code assigned to identify a transaction.
al_NewTrID	Long	4	The ID of the average priced trade.
al_OrigTrID	Long	4	For standard trades this argument is zero, otherwise for take-up trades this argument contains the al_TrID

			corresponding to the trade that was given-up.
al_OtherMbr	Long	4	A code that identifies another participant who is associated with the particular transaction, e.g. the other party to a trade.
al_OtherMbrID	Long	4	A number assigned to a participant that uniquely identifies that participant.
al_PositionLines	Long	4	The number of lines in a transfer batch, excluding support info lines i.e. the number of position lines plus one for the batch header.
al_PriceAvgID	Long	4	A unique ID assigned to each Price Averaging request.
al_Qty	Long	4	The quantity associated with the particular transaction. Note that quantity is an absolute value for some transactions and a relative value for other transactions.
al_qtyMatchedOut	Long	4	The quantity originally matched out.
al_QtyReOpen	Long	4	The quantity to be reopened.
al_ReqSeq	Long	4	A number that uniquely identifies a request for a transfer for a particular item of trading activity.
al_SettPrice	Long	4	The cash settlement price. This price is unformatted and does not contain any decimal places.
al_StatusCheckSeq	Long	4	Uniquely identifies a status check (echo) request issued by either the participant or by ASXCL. In the case of status check requests issued by the participant, the status check sequence number should be one greater than the previously issued status check sequence number.
al_Strike	Long	4	The strike price associated with the traded entity. Note that this price is unformatted and does not contain any decimal places.
al_SupportLines	Long	4	The number of supporting information messages in the transfer batch.
al_TenderStyle	Long	4	Reserved for future use.

al_ToAccID	Long	4	The al_AccID of an account that subject to the approval of a transfer has contracts transferred into its open position.
al_ToMbrID	Long	4	The number identifying a participant that subject to the approval of a transfer has contracts transferred into its open position.
al_TransID	Long	4	A number assigned to a transfer batch that uniquely identifies that transfer batch. Numbering must start at one and be incremented by one for each transfer batch.
al_TrID	Long	4	A number assigned to a trade which uniquely identifies that trade.
al_Units	Long	4	The number of units of collateral or units held in the ASX Commodity Depository System.
al_Version	Long	4	The version number associated with the traded entity.
as_Acc	String	10	The code that is commonly used to identify an account and must be upper case. The first character of the account code must be in the range of zero to nine, or A-Z.
as_AcceptedBy	String	1	A code indicating who has accepted the transfer. Valid values are: M = Participant C = ASXCL.
as_AcceptFlag	String	1	Used to indicate whether or not a trade has been given-up or has been accepted or rejected by the other party; or deleted (rejected) by ASXCL. Valid values are: N = No; rejected Y = Yes; accepted D = Deleted (by ASXCL).
as_AcceptReject	String	1	Used to indicate whether an account name change has been accepted or rejected by ASXCL. Valid values are: Y = Accepted N = Rejected.
as_AccName	String	250	The name of an account.

as_AccNameConfirm	String	1	Used to indicate whether or not confirmation of an account name change is expected by ASXCL. Valid values are: Y = Yes N = No.
as_AccStat	String	1	A code indicating the status of an account. Valid values are: A = Active I = Inactive S = Suspended.
as_AccType	String	1	The account type. Valid values are: H = House account C = Consolidated account P = Private account I = Institution R = RIOT T = RT. Note: Accounts types C, R and T can only be assigned by ASXCL.
as_ActDesc	String	20	A description of an activity.
as_Active	String	1	Used to indicate the trading status of the participant. Valid values are: Y = Yes N = No.
as_ActType	String	1	The activity type for reserved cash messages. Valid values are: L = Lodgement W = Withdrawal
as_Address1	String	50	Address line 1.
as_Address2	String	50	Address line 2.
as_Address3	String	50	Address line 3.
as_Address4	String	50	Address line 4.
as_AllowExer	String	1	If set to Y, exercises are currently allowed for this product.
as_AllocRef	String	50	Reference information entered by user when allocating contracts. The reference is comma separated.
as_AmendmentType	String	1	A code indicating the type of amendment contained within a message. Valid values are:

			N = New record E = Edit to an existing record D = Deletion of an existing record. Note: Some methods do not permit all three values. Refer to the appropriate method to determine whether there are any overriding notes.
as_AttachName	String	255	The name and location of an attachment to a message sent by ASXCL.
as_AttachType	String	50	The type of attachment, for example a Word document, Excel spreadsheet, etc.
as_AutoExer	String	1	A code that indicates whether an exercise notice should be automatically generated for an account on the expiry date. Valid values are: N = No - don't generate exercise notices Y = Yes - generate exercise notices if other criteria are satisfied.
as_AutoMatchOut	String	1	A code indicating whether or not long and short positions for the same Traded Entity should be automatically liquidated to the fullest extent possible. Valid values are: N = No - don't liquidate offsetting positions Y = Yes - automatically liquidate offsetting positions.
as_AvgPriceText	String	10	Reserved for future use.
as_Basket	String	1	Indicates whether the derivative product is a basket. Valid values are: N = Not a basket T = True style basket, i.e. the components of the basket are true underlying products. S = Synthetic style basket, i.e. the components of the basket are not 'true' underlying products.
as_BCastText	String	255	The content of a message sent by ASXCL.

as_BCastTitle	String	80	The title of a message sent by ASXCL.
as_BCastType	String	2	The type of message. Valid values are: C = Critical I = Informational W = Warning.
as_BuySell	String	1	Indicates whether the transaction relates to a long/bought or short/sold position. Valid values are: B = Bought S = Sold.
as_CapAdj	String	1	'Y' indicates the entity has been the subject of a corporate action; or else 'N'.
as_CashWithdrawals	String	1	A code indicating whether cash withdrawal requests are accepted by ASXCL. Valid values are: N = No - transactions of this type are not accepted. Y = Yes - transactions of this type are accepted.
as_ChargeGST	String	1	Indicated whether GST is to be charged. Valid values are: Y = Yes, charge GST N = No, do not charge GST.
as_CollActType	String	20	Code that identifies the type of activity performed. Possible values are: "ADD" "RE-ISSUE" "TRANSFER OUT" "TRANSFER IN" "WITHDRAWN" "DELETE BONUS".
as_CollDetail	String	50	Provides further details relating to collateral, e.g. Share class, Bank Bill series etc.
as_CollType	String	2	Code identifying the type of collateral. Valid values are: AC = Austraclear

			BG = Bank Guarantee S = Shares.
as_CommBasis	String	1	Indicates the basis of the commission value. Valid values are: A = amount R = rate (per lot) P = percentage (of the deal value).
as_Comment	String	80	The comment which accompanies a transfer through the various stages through to its completion.
as_CompType	String	1	Composite type used to identify composite trades. This field is empty for standard trades. A valid value is T = Time Spread.
as_ConditionCodes	String	16	ASX standard codes that define attributes of a trade.
as_Contra	String	1	A code indicating whether or not the trade is a Contra trade input by ASXCL to reverse a previous erroneous trade record. Valid values are: N = No - not a Contra trade Y = Yes - Contra trade.
as_CoverGrp	String	20	The code that is used to identify the collateral cover group. *#N/A#* indicates no cover group is applicable.
as_CPRef	String	250	Additional information supplied by a participant for use by that particular participant.
as_Cur	String	3	Uniquely identifies a currency.
as_CHStatus	String	1	Indicates the status of consideration given by ASXCL. 'P' indicates 'Pending', 'Y' indicates 'Approved' and 'N' indicates 'Rejected'.
as_CHReason	String	255	Narrative supplied by ASX to explain its action.
as_DelMonthText	String	7	The delivery month in text format for listed traded entities is in MMMYY format. For OTCs, the expiry date is usually non-standard in the format of DDMMMYY. This field allows OTCs

			to be easily recognisable in listing and enquiries.
as_DelProd	String	6	A code used to identify items within the ASX Commodity Depository System that have similar characteristics, i.e. they are the same commodity.
as_DepositID	String	20	A unique number assigned to identify a particular item held in the ASX Commodity Depository System.
as_DerivProd	String	6	The code that is commonly used to identify a derivative product. It must be upper case.
as_DerivProdDesc	String	60	The description of a derivative product.
as_DerivProdStat	String	1	A code indicating the status of a derivative product. Valid values are: A = Active S = Suspended.
as_DerivProdType	String	2	A code indicating the type of derivative product. Valid values are: FU = Future OF = Option on Future OI = Option on Index LE = Low Exercise Price Option (LEPO) OS = Equity Option S = Stock SI = Stock Index.
as_Description	String	255	The description of a file transferred by ASXCL.
as_DefRT	String	1	Indicates whether the account is the default account of an RT/RIOT. Valid values are: N = No, this account is not the default Y = Yes, this account is the default.
as_Direction	String	1	A code indicating whether a transfer is transferring positions into an account or is transferring positions out of an account. Valid values are: I = in; transferring positions in O = out; transferring positions out.

as_EntityShortCut	String	8	The code that is commonly used to identify a traded entity and it must be upper case.
as_EqtClear	String	4	The code that is commonly used to identify an Equity Clearer – SEATS number.
as_EqtClearName	String	50	The name of the Equity Clearer.
as_ExchRef	String	10	The code that is commonly used to identify a trade (both bought and sold components).
as_Exercise	String	1	A code indicating whether transactions relating to exercises or positions that are due to expire at the end of day are accepted by ASXCL. Valid values are: N = No; transactions of this kind are not accepted Y = Yes; transactions of this kind are accepted.
as_ExerStyle	String	1	Indicates the exercise style associated with the traded entity. Valid values are: A = American E = European.
as_FailedGiveUp	String	1	'Y' indicates a failed give-up or else it is 'N'.
as_Final	String	1	Valid values are as follows: 'Y' indicates the invoice is final 'P' indicates the invoice is provisional.
as_FileName	String	255	The location and name of the file that is awaiting collection from ASXCL. Note that as a security measure, the location of files is not predictable.
as_FileType	String	1	A code indicating the type of file. Valid values are: T = Text File R = Report File (.rpt) X = XML file.
as_Frequenct	String	1	A code indicating the frequency the report is to be printed. Valid values are:

			D = Daily F = Weekly – First Day L = Monthly – Last Day M = Monthly – First Day W = Weekly - Last day.
as_FromMbrReason	String	255	Justification supplied by a participant.
as_GiveUp	String	1	A code indicating whether give-up transactions are accepted by ASXCL. Valid values are: N = No; transactions of this type are not accepted Y = Yes; transactions of this type are accepted.
as_GroupLevel	String	1	A code indicating whether the account or Cover Group holds the collateral. Valid values are: N = Account Y = Cover Group.
as_Grp	String	10	The code that is commonly used to identify an account group that must be upper case.
as_GrpName	String	50	The name of an account group.
as_GUCommBasis	String	1	Indicates the basis of the commission value in respect of give-ups. Valid values are: R = rate (per lot) P = percentage (of the deal value).
as_HeldAt	String	10	Code that identifies the location where a commodity is stored.
as_Holder	String	100	The registered holder of the collateral.
as_InOut	String	1	A code indicating whether a transfer is transferring positions into an account or is transferring positions out of an account. Valid values are: I = in - transferring positions in O = out - transferring positions out.
as_IsOTC	String	1	A code indicating whether or not the traded entity is an OTC or a listed traded entity. Valid values include: N = No; not an OTC (i.e. Listed)

			Y = Yes; OTC.
as_IssuerCode	String	6	A code used to identify the issuer of the collateral.
as_Ledger	String	12	A code that identifies the ledger for a financial transaction.
as_LodgeID	String	12	The code uniquely identifying the collateral lodgement.
as_LogText	String	255	Text relating to a reconciliation discrepancy.
as_LogType	String	1	A code indicating the type of reconciliation logs entry. Valid values are: S = Start M = Maximum discrepancies exceeded R = Reconciled D = Discrepancy Z = Summary.
as_ManAutoExer	String	1	If set to Y, all positions for the derivative product are included in the auto-exercise process on expiry day regardless of the auto-exercise indicator for the accounts. If set to N, only positions where auto-exercise indicator for the account is Yes are included in the auto-exercise process on expiry day.
as_Mbr	String	4	The mnemonic used to identify a participant.
as_MbrClearType	String	1	A code indicating the clearing participant type of the participant. Valid values are: G = General Clearing participant R = Market Maker N = Non-Clearing participant.
as_MbrInfo	String	50	Optional free format text for participant use only.
as_MbrName	String	60	The name of the participant.
as_MbrShortName	String	20	A shortened form of a participant's name.
as_MoreDetails	String	250	Information that describes a commodity.

as_MktMaker	String	1	A code indicating whether the trade was executed by a Market Maker. Valid values are: I = Executed by a RIOT R = Executed by an RT blank = not executed by a Market Maker.
as_MsgDatabaseName	String	255	The name and location of the MCM message database PA_MCMM. The location of this database is determined at the time the MCM is installed.
as_MsgQueue	String	64	Uniquely identifies a message queue used in the transfer of messages between ASXCL and the participant site.
as_MsgStartEnd	String	1	A code indicating the position of the message within a message set. Valid values are: Empty = Not part of a message set E = Last message in a set of messages M = Message between the start and end of a message set S = First message in a message set.
as_MsgTag	String	255	This tag is attached for message rejections to provide further information about the original (rejected) message. The format is: <tag id>=<tag text><tag separator>:<tag id>=<tag text>… where <tag separator> is ,; and <tag id> has the following values: MT = Message type PA = Price average batch ID TR = Trade ID AL = Allocation sequence number TE = Traded entity ID AC = Account ID TO = Transfer batch origin TI = Transfer batch ID AT = Amended type GI = Group ID DT = DerivProd type MI = Participant ID CR = Currency

			BC = Broadcast ID UI = User ID DM = Matchout date SG = SegType.
as_MsgText	String	1020	The actual content of a message received from ASXCL.
as_MsgType	String	2	A code used to identify the type of message sent to or received from ASXCL.
as_OpenClose	String	1	Reserved for future use.
as_OptType	String	1	Indicates whether the traded entity relates to an option, and if so, the type of option. Valid values are: C = Call P = Put.
as_Origin (Trade Related)	String	1	A code indicating the origin of the trade. Valid values are: A = Adjustment, a trade entered by ASXCL G = Given-up, a trade given-up by another clearing participant T = Traded, a standard trade.
as_Origin (transfer related)	String	4	The participant mnemonic of the transferor. For Bulk Transfers the mnemonic is CH for ASXCL.
as_PaperlessColl	String	1	'Y' indicates that the account has provided permission to lodge collateral without requiring a signature, or else it is 'N'.
as_PhysRef	String	20	A unique number assigned to identify a particular tender.
as_Produce	String	1	A code indicating whether to ASXCL is to produce the report or not. Valid values are: N = No; the report is not required Y = Yes; the report is required.
as_RejReason	String	255	The reason supplied by a participant for rejecting a give-up, or requesting an undo give-up.
as_ReconcileID	String	30	The code that is used to identify a type of reconciliation. It must

			correspond to a reconciliation type accepted by ASXCL.
as_RepID	String	20	The code that is used to identify a report.
as_Ref	String	15	A value passed to ASXCL by an exchange and associated with a particular trade.
as_RefreshType	String	1	A code indicating the type of refresh. A valid value is R = Refresh, a standard refresh.
as_RejectedBy	String	1	A code indicating the party responsible for rejecting a transfer. Valid values are: M = Participant C = ASXCL.
as_RequestedAmt	Currency	8	The requested withdrawal amount sent to ASXCL.
as_RoundType	String	1	Reserved for future use.
as_SecCode	String	3	Underlying security code.
as_SegType	String	1	A code indicating whether an account is a segregated or unsegregated account. Valid values are: S = Segregated U = Unsegregated.
as_SettPriceText	String	10	The formatted cash settlement price. It contains the appropriate number of decimal places for the derivative product.
as_ShutDown	String	1	A code indicating whether ASXCL will accept any transactions. Valid values are: N = No, transactions are not accepted Y = Yes, transactions are accepted.
as_SoftVer	String	10	The version number of the MCM software.
as_SpecificCover	String	1	Indicates whether shares lodged by the account as collateral are to be treated as specific cover.
as_Spot	String	1	A code indicating whether or not the traded entity is a spot or not. Valid values include: N = No, not spot

			Y = Yes, spot.
as_StrikeText	String	10	The formatted strike price associated with the traded entity. It contains the appropriate number of decimal places for a derivative product.
as_SupportInfo	String	100	This information is used to support the transfer request, e.g. of date of trade, traded price etc.
as_TakeUp	String	1	A code indicating whether take-up transactions are accepted by ASXCL. Valid values are: N = No, transactions of this kind are not accepted Y = Yes, transactions of this kind are accepted.
as_TenderDetails	String	250	Narrative that provides an explanation of the tender value.
as_ToAcc	String	10	The code that identifies the account to which positions are to be transferred.
as_TonightOnly	String	1	A code indicating whether the request applies to that day's overnighot process only or not. Valid values are: N = No; the report is not required Y = Yes; the report is required for tonight only.
as_Trader	String	10	The code that is used to identify the participant who executed the trade.
as_TraderType	String	1	The code that is used to identify the type of trader who executed the trade.
as_TradingCeased	String	1	A code indicating whether the entity is no longer traded. Valid values are: N = No, traded Y = Yes, not traded.
as_TrDerivProd	String	6	A code that identifies a derivatives product.
as_TrEntDesc	String	40	A description of the traded entity.
al_TransferQty	Long	4	The number of contracts that the transaction concerns.
ac_TransferComm	Currency	8	The commission value associated with the transaction.

as_TransferAccept	String	1	A code indicating whether accept/reject transactions are accepted by ASXCL. Valid values are: N = No; transactions of this kind will not be accepted Y = Yes; transactions of this kind will be accepted.
as_TransferEntry	String	1	A code indicating whether or not the entry of new transfers is accepted by ASXCL. Valid values are: N = No, transactions of this kind are not accepted Y = Yes, transactions of this kind are accepted.
as_TrPriceText	String	10	The traded price formatted so that it contains the number of decimal places associated with the particular derivative product.
as_TrOrderNo	String	17	The number assigned by the Derivatives Trading Facility to identify an order that gives rise to the particular trade.
as_TUCommBasis	String	1	Indicates the basis of the commission value in respect of take-ups. Valid values are: R = Rate (per lot) P = Percentage (of the deal value).
as_UnitCode	String	6	The code defining a collateral unit. A valid value is the appropriate currency code “Shares”.
as_URLName	String	255	The location and name of the file that is awaiting collection from ASXCL. Note that as a security measure, URL names cannot be predicted.
as_UserID	String	10	The ID of the user that performed the transaction. Note that this parameter is used for reference purposes only and is not validated by the clearing system.
as_VersionText	String	3	The formatted version number associated with the traded entity.

awrk_MsgWorkspace	Workspace	N/A	A reference to the workspace in which the database object required for messaging is to be created. The database object should be a participant of the same workspace, i.e., responsible for the update that generated a message for transmission to ASXCL, or for processing a message received from ASXCL.

4. Error Handling

4.1. Error Handling in ActiveX Controls

Messages sent by a participant are subject to initial validation performed by the SendMsg ActiveX control prior to acceptance of the message for transmission. Following the transmission of a message by ASXCL, it is then subject to more stringent validation. As a consequence, messages may be rejected by the SendMsg ActiveX control or by ASXCL.

The SendMsg ActiveX control validates that the message can be written to the message database for subsequent transmissions.

All methods of the SendMsg ActiveX control return a Boolean value that indicates whether or not the call was successful. When a value of False is returned, the two public properties ErrorNum and ErrorDesc are populated with a reference number and a description of the particular error that will enable the cause to be identified.

4.2. Rejection Messages

All messages received by ASXCL are subjected to stringent validation. Should the ASXCL reject a message, then a message is issued (using the GetBCast_V1 method) providing a brief explanation as to why the message was rejected. The message issued by ASXCL also identifies the sequence number of the message that was rejected.

Important

ASXCL monitors the number of messages sent by each participant that have been rejected. If the number of rejections for a participant exceeds a limit established by ASXCL, then communications with the particular participant are terminated until the cause has been investigated and the count of messages rejected has been reset.

The following table provides a list of the error rejection messages that may be transmitted by ASXCL.

Note:

At the time a message is issued, the characters '%d' and '%s' shown below will be replaced with data values that assist in identifying the cause of the rejection.

Error ID	Error Message	Explanation
50002	Take-up has already been accepted/rejected.	An attempt has been made to alter the status of a take-up that has already been accepted or rejected.
50003	This trade does not relate to a take-up.	The take-up status of a trade cannot be altered if the trade is not a take-up.

50004	Unknown Amendment Type.	The amendment type received is not recognised by ASXCL.
50005	The quantity to be allocated exceeds the unallocated quantity associated with the trade.	Self-explanatory.
50006	Unable to locate bought/sold open positions with sufficient quantity to satisfy this match out.	Self-explanatory.
50007	An account with AccID %d already exists.	An attempt has been made to insert a new account with an AccID that is already in use.
50008	There is no account with AccID %d.	The message received by ASXCL referred to an AccID that does not exist.
50011	There is no trade with TrID %d.	The message received by ASXCL referred to a TrID that does not exist.
50012	An allocation with AllocSeq %d already exists.	An attempt has been made to insert a new allocation/give-up with an AllocSeq that has already been used.
50013	There is insufficient quantity open (%d contracts only) to enable the undo of this allocation.	Self-explanatory.
50014	There is no allocation with AllocSeq %d.	An attempt has been made to remove an allocation/give-up with an AllocSeq that does not exist.
50015	There is no Participant with MbrID %d.	The message received by ASXCL referred to an MbrID that does not exist.
50016	You cannot Give-up to a Non-Clearing participant.	Self-explanatory.
50017	This account cannot be deleted as it has an open position.	Self-explanatory.
50018	Account %s is already in use.	An attempt has been made to insert a new account with an account code that is already in use.
50019	A group with GrpID %d already exists.	An attempt has been made to insert a new account group with a GrpID that is already in use.
50020	Group %s is already in use.	An attempt has been made to insert a new account group with a Group code that is already in use.
50021	There is no group with GrpID %d.	The message received by ASXCL referred to a GrpID that does not exist.
50022	AccID %d is already linked to GrpID %d.	An attempt has been made to link an account to an account group where it is already linked.
50023	AccID %d is not a participant of GrpID %d.	An attempt has been made to remove an account from its membership of an account group; however the specified account is not a participant of the particular group.

50024	There is no Derivative Product with DerivProd %s.	The message received by ASXCL referred to a DerivProd that does not exist.
50025	There is no Commission Rate for MbrID %d, DerivProdType %s and Currency %s.	An attempt has been made to change or remove commission rates associated with an MbrID/DerivProd combination; however the rates do not exist for the particular MbrID/DerivProd combination.
50026	Commission rates for MbrID %d and DerivProd %s are already defined.	An attempt has been made to insert commission rates associated with an MbrID/DerivProd combination; however rates for this combination have already been inserted.
50027	Trade %d has been deleted.	An attempt has been made to update a trade that is flagged as deleted.
50028	Unable to locate %s open positions.	An attempt has been made to adjust an open position that does not exist.
50030	The transfer quantity exceeds the quantity available.	Self-explanatory.
50032	Unable to locate Broadcast Message ID %d for Participant ID %d.	The message received by ASXCL refers to a message that was not sent by ASXCL.
50033	The schedule %s has already been deleted.	An attempt has been made to delete an exercise schedule that does not exist.
50034	The schedule %s already exists.	An attempt has been made to insert an exercise schedule that already exists.
50035	The schedule %s does not exist.	An attempt has been made to update an exercise schedule that does not exist.
50036	The product %s does not exist.	The message received by ASXCL referred to a product that does not exist.
50037	The %s reconcile ID does not exist.	The message received by ASXCL referred to a Reconciliation Identifier that the ASXCL does not recognise.
50038	The Status Check Sequence %d does not exist.	An attempt has been made to update a status check that was not sent by ASXCL.
50039	Current available quantity has changed.	The Exercise/Exclude from Exercise message received by ASXCL attempted to adjust the open position when there was an insufficient quantity available.
50040	The position does not exist.	The message received by ASXCL attempted to adjust an open position that does not exist.
50042	The exchange ID (%d) supplied does not correspond to the exchange ID (%d) associated with the traded entity.	Self-explanatory.
50043	There is no entity with EntID %d.	The message received by ASXCL referred to an EntID that does not exist.
50050	Exercise schedule %s cannot be deleted as it is currently in use.	Self-explanatory.

50051	This position does not relate to an option.	Self-explanatory.
50052	There is no Price Averaging batch with ID %d.	Self-explanatory.
50053	Price Averaging Trade %d has been allocated or given-up. The allocation or give-up has to be reversed before the Price Averaging batch can be undone.	An attempt has been made to undo a Price Averaging request but the resultant Price Average trade associated with the request has been allocated or given-up.
50054	Trades relating to take-ups cannot be part of the Price Averaging batch.	Self-explanatory.
50055	Only fully unallocated trades can be part of a Price Averaging batch.	Self-explanatory.
50056	Trade %d is associated with traded entity %d, not %d, as required by the Price Averaging Batch.	The trade, which is included as part of the Price Averaging request relates to a different, traded entity to that specified on the Price Averaging header message.
50057	The Buy/Sell indicator %s associated with trade %d does not match the Buy/Sell indicator %s required by the Price Averaging batch.	The trade that is included as part of the Price Averaging request has a different buy/sell indicator to that specified on the Price Averaging header message.
50058	Price Averaging batch %d is already in use.	Self-explanatory.
50059	Account: %s – change to the segregation type disallowed because the account belongs to a cover group.	Self-explanatory.
50060	This product is subject to conditional trading. %s are not permitted.	Matchouts and exercise instructions are not permitted for derivative products under conditional trading.
50061	There is no Derivative Product Type %s.	Self-explanatory.
50062	There is no currency %s.	Self-explanatory.
50063	This account cannot be deleted as it is a default RIOT account.	An attempt has been made to delete an account, which is the default account for a RIOT.
50064	This account cannot be deleted as collateral has been lodged for it.	Self-explanatory.
50065	This account cannot be deleted as at least one transfer involving this account is pending.	An attempt has been made to delete an account which is involved in a transfer that is still pending.
50067	Give-up of this trade is not permitted.	An attempt was made to give-up a trade executed by a RT or RIOT.
50068	This trade must be allocated to a %s account associated with the %s that executed the trade.	An attempt was made to allocate a trade executed by a RT or RIOT to an unrelated account.
50069	This account cannot be deleted. It is the default account for %s %s.	Self-explanatory.

50072	The value of commission supplied does not match the commission basis parameters.	The value of commission calculated by ASXCL using the commission basis parameters supplied does not match the commission amount supplied.
50074	This account code cannot be modified as the account has %s.	Amendment of the account code is not permitted because the account may have an open position etc.
51001	Unsupported message type %s and version %d. %s and %d contains the message type and version number associated with the message rejected by ASXCL.	ASXCL does not recognise the message type and/or version number of the message received.
51002	Message number %d is corrupt. Packer failed with %s. %s contains the error message output by the Data Compression facility while trying to decompress a message.	ASXCL is unable to decompress the message received.
51003	%s must be zero or greater.	Self-explanatory.
51004	A space is not allowed in %s.	Self-explanatory.
51005	A single quote is not allowed in %s.	Self-explanatory.
51006	A tab is not allowed in %s.	Self-explanatory.
51007	%s has exceeded the maximum allowable length of %s.	Self-explanatory.
51008	Exercise Cut-off time has been reached.	Self-explanatory.
51009	Give-up Cut-off time has been reached.	Self-explanatory.
51010	Take-up Cut-off time has been reached.	Self-explanatory.
51011	Transfer Accept/Reject Cut-off time has been reached.	Self-explanatory.
51012	Transfer Entry Cut-off time has been reached.	Self-explanatory.
51013	%s must be either "Y" or "N".	Self-explanatory.
51014	There is no trade with TrID %s.	The message received by ASXCL referred to a TrID that does not exist.
51015	Mandatory field %s missing.	Self-explanatory.
51016	%s is invalid.	Self-explanatory.
51017	This transfer has already been accepted.	Self-explanatory.
51018	This transfer has already been rejected.	Self-explanatory.
51019	Transfer acceptance received from an invalid participant.	Self-explanatory.
51020	Transfer rejection received from an invalid participant.	Self-explanatory.

51021	Invalid Amendment Type indicator - it must be "N", "E" or "D".	Self-explanatory.
51022	Invalid Segregation Type. Must be "S"egregated or "U"nsegregated.	Self-explanatory.
51023	Invalid Account Status.	Self-explanatory.
51024	Invalid Log Type.	Self-explanatory.
51025	The transferor participant ID is invalid.	Self-explanatory.
51026	The transferor participant ID is invalid.	Self-explanatory.
51027	The transferee's account is invalid.	Self-explanatory.
51028	%s cannot be negative.	Self-explanatory.
51029	A vertical bar is not permitted in %s.	Self-explanatory.
51030	Either “O”pening or “C”losing must be specified.	Reserved for future use.
51031	The number of transfer lines supplied does not match the batch header.	For the GetCHTransferSupportLine_V1, the error occurs if the numbers of batch lines or support lines do not match the values in the header.
51032	The transferee participant ID is invalid.	Self-explanatory.
51035	%s must not contain more than %s decimal places.	Self-explanatory.
51036	%s is greater than permitted.	Self-explanatory.
51038	The text price of %s does not match the numeric price of %s.	Self-explanatory.
51039	The exercise cut-off time is yet to be reached. A response to an exercise cut-off can only be sent after the exercise cut-off time has been reached.	Self-explanatory.
51040	A response to an exercise cut-off has already been received.	Self-explanatory.
51041	An automatic exercise schedule can only be supplied when the account is set to automatically exercise.	Self-explanatory.
51042	%s cannot be greater than %s.	Self-explanatory.
51043	%s cannot be later than the previous business date.	Self-explanatory.
51044	The %s cut-off time has been reached. You cannot submit this undo match-out request.	Self-explanatory.
51045	The first character of the account code must be in the range of zero to nine or A-Z.	Self-explanatory.

51046	The Transferor's participant ID is invalid.	Self-explanatory.
51047	The Transferee's participant ID is invalid.	Self-explanatory.
51048	The Transferor's account is invalid.	Self-explanatory.
51049	The Transferee's account is invalid.	Self-explanatory.
51050	Invalid segregation type. This segregation type is not valid for the Account Type.	Self-explanatory.
51051	The average price received %s does not match the average price %s calculated by the system.	Self-explanatory.
51052	Either R for rate, P for percentage or A for amount must be specified.	Self-explanatory.
51053	Exercise Cut-off time has been reached.	Self-explanatory.
51054	Deprecated.	Deprecated.
51055	Cash withdrawals cut-off time has been reached.	Self-explanatory.
51056	%s must be greater than zero.	Self-explanatory.
51057	Cash withdrawals are not permitted.	Self-explanatory.

The following are some of the possible underlying causes for a message being rejected by ASXCL:

- Details contained in the message are incorrect
- The Participant's database is not in-line with the database maintained by ASXCL
- An incorrect version of the SendMsg ActiveX control is being used
- The message has been corrupted either in the message database or during transmission to ASXCL.

5. Method to Message Type Cross Reference

The following table provides a summary of the methods available on both SendMsg and RecMsg ActiveX controls. It maps the message type associated with each method.

Method	Message Type
SendMsg ActiveX Control	
SendAcc_V1, SendAcc_V2	AC
SendAccGrp_V1	AG
SendAccGrpMembership_V1	AM
SendAlloc_V1	AL
SendBCastViewed_V1	BV
SendCashWithdrawals_V1	CW
SendCHStatusCheckResponse_V1	SM
SendCommRate_V1	CR

SendEODRepRequest_V1	ER
SendExerciseExclude_V1	XX
SendExerciseManual_V1	XE
SendExerciseResponse_V1	XR
SendGiveUp_V1	GU
SendGiveUpUndoRequest_V1	GR
SendMatchOut_V1	MO
SendPriceAvgHead_V1	PH
SendPriceAvgLine_V1	PL
SendPriceAvgUndo_V1	PU
SendReconLog_V1	RL
SendRestartMbrProc_V1	RM
SendStatusCheck_V1	SC
SendTakeUp_V1	TU
SendTRActTransferAccept_V1	TY
SendTRActTransferReject_V1	TN
SendTRActTransferRequest_V1	TF
SendTransferHead_V1	TH
SendTransferLine_V1	TL
SendTransferMbrAccept_V1	TA
SendTransferMbrHead_V1	MH
SendTransferMbrLine_V1	ML
SendTransferMbrReject_V1	TJ
SendTransferSupportLine_V1	TS
SendUndoAlloc_V1	UA
SendUndoMatchOutRequest_V1	UM
RecMsg ActiveX Control	
GetBCast_V1	BC
GetBCast_V1 (Mail)	MA
GetBulkTransferHead_V1	BH
GetBulkTransferLine_V1	BL
GetBuyTenderDetails_V1	DD
GetCHAcc_V1	AH
GetCHAccName_V1	AN
GetCHAlloc_V1	AA

GetCHGiveUp_V1	CG
GetCHStatusCHeck_V1	ST
GetCHTrActTransferAccept_V1	CY
GetCHTrActTransferRequest_V1	CF
GetCHTransferAccept_V1	CA
GetCHTrActTransferReject_V1	CN
GetCHTransferMbrHead_V1	CH
GetCHTransferMbrLine_V1	CL
GetCHTransferSupportLine_V1	TS
GetCHTransferReject_V1	CJ
GetCollateralActivity_V1	CM
GetDerivProd_V2	DP
GetDepositoryActivity_V1	HA
GetEquityClearer_V!	EQ
GetFacility_V1	FC
GetFileTransfer_V1	FT
GetGUAdvice_V1	GA
GetGUDeletion_V1	GD
GetMarketMakerObligations_V1	MK
GetMbr_V1	MB
GetMembership_V1	MM
GetPriceAvgAct_V1	PD
GetPosAdj_V1	PA
GetRefreshEnd_V1	RE
GetRefreshStart_V1	RS
GetSoftwareUpgrade_V1	SU
GetStartNewDayEnd_V1	SE
GetStartNewDayStart_V1	SS
GetStatusCheckResponse_V1	SR
GetTenderDetails_V1	DD
GetTenderAdvice_V1	DA
GetTrade_V1	TR
GetTradeDeletion_V1	TD
GetTradedEntity_V1	TE
GetTransferConf_V1	TC

GetUndoMatchOutRequestRej_V1	UJ
Intra-day Capital Adjustment message (See note below)	IC
Intra-day Reference Price message (See note below)	IR
Intra-day Theoretical Price message (See Note below)	IT
Restart Processing message (See Note below)	RP
The Update Reference Price message (See Note below)	UR

Note:

Message types IC, IR, IT, RP and UR do not contain any data, and therefore no methods are required for retrieval of the arguments associated with the messages.

6. Appendix 1 - Glossary

Term	Definition
ActiveX Control	An ActiveX control is a software framework created by Microsoft that enables content to be downloaded via the internet.
Argument	An argument is a value that is passed between programs, subroutines or functions. They are independent items or variables that contain data or codes.
Calling Application	Client application.
Low Exercise Price Option (LEPO)	A call option with an exercise price of one cent and an agreement to purchase 1000shares. This avoids paying stamp duty and enables the investor to gain most of the features of owning the share except dividends and voting rights.
Method	A method is a subroutine (or procedure) that defines the behaviour to be exhibited, and uses arguments to perform the method.
Message	A message is used to invoke a particular method.
Message Set	A set of related messages.
Message Type	The message type is the code associated with a particular message.
Registered Independent Option Trader (RIOT)	RIOT accounts must have an account type of either Institution or Private and therefore must be a segregated account. Trades executed by RIOTs must be allocated to a RIOT account associated with that RIOT.

Registered Trader (RT)	RT accounts must be a House account. Trades executed by RTs must be allocated to a RIOT account associated with that RT.
Transaction	A discrete business process that will either fully succeed or fully fail.
Transmission	The physical sending of data from one system to another.