Additions to the Ariel API

Author: *Mike Woolley <mike@arielcommunications.co.uk>*
Date: *20th October*
Version: *1.5 (for API v4.00)*

Introduction

This document describes new Additions to the Ariel API. Only changes are presented here, for existing functionality, please see the main API specification.

Open questions and items still requiring specification are shown like this [*Italics*].

Table of Contents

Chat

Summary of Chat Functions, Properties & Events

Func tions	
Chat Start	Start a new Chat.
Chat End	Finish a Chat.
Chat Chars	Send Chat characters as they are typed.
Chat Send	Send Chat message and transfer control to the dealer.
Chat Interrupt	Interrupt the dealer and take control of the Chat.

Functions	
Chat Interrupt Done	Acknowledge dealer interrupt and transfer control to the dealer.

Properties	
Accept Reverse Chat	Accept or reject "Reverse Chat" requests.

Events	
Chat Accepted	Dealer has accepted or declined to Chat.
Chat Chars	Characters from the dealer as they are typed.
Chat Send	Message from dealer and transfer of control to the client.
Chat Interrupt Done	Dealer has acknowledged client interrupt.
Chat End	Dealer has ended the Chat.
Reverse Chat Request	Dealer has requested a Chat with the client.
Dealer Interrupt	Dealer has requested an interrupt (described in main document).
Dealer Held	The dealer has put a request on hold (described in main document).
Converted To Chat	The current deal has been converted to a Chat session by the dealer.

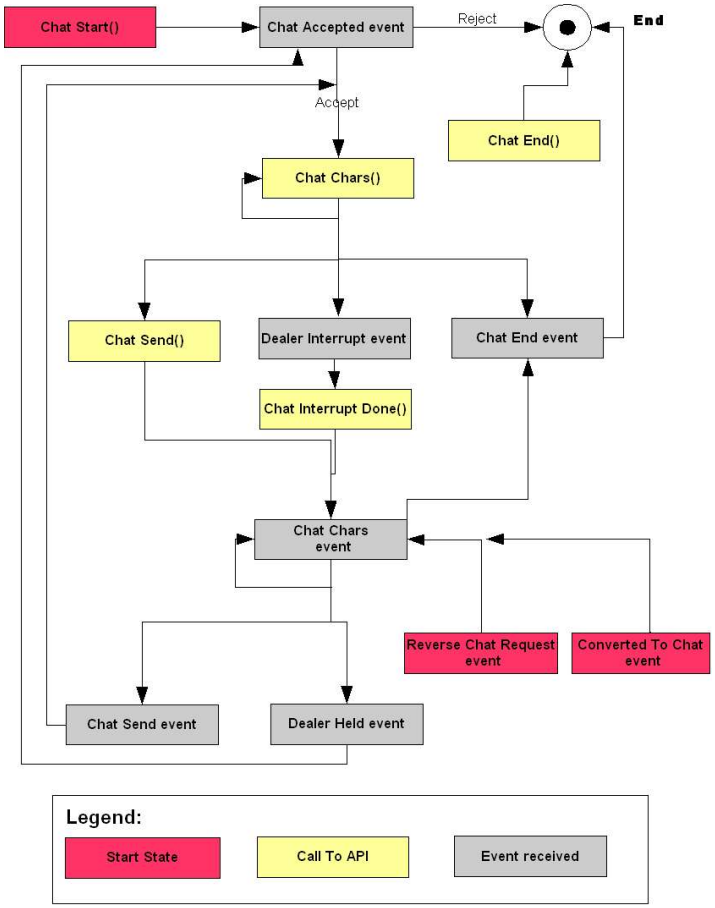
Flow of Chat Functions & Events

Chat requests are usually initiated by the client. Conversations are "half-duplex", ie only one side can talk at once. Characters are sent to the other party as soon as they are typed (via the **ChatChars** function or event). The conversation changes sides when the party that is talking hits the "Send" button (indicated by the **ChatSend** function or event). The party that is listening can switch the conversation to their side by hitting the "Interrupt" button (via the **ChatInterrupt** function or **DealerInterrupt** event). Either party can terminate the conversation at any time by hitting the "End" button (indicated by the **ChatEnd** function or event).

The dealer can also request a conversation with the client ("Reverse Chat"). This is indicated by receipt of the **ReverseChatRequest** event. Reverse Chat requests will only be accepted if the **AcceptReverseChat** property is true (non-zero).

In addition, a deal in progress can be converted to a Chat sessions by the dealer. This is indicated by receipt of the **ConvertedToChat** event.

Here is a diagram showing the flow of the various functions and events in more detail:



Chat Start Function

Start a new Chat..

Parameters		
String	**Session Id**	
String	**Message**	Initial message for dealer

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.
String	**Request ID**	

Errors	
22	**Not logged in**
2	**Invalid session Id**

Chat End Function

End a Chat.

Parameters		
String	**Session Id**	

Parameters		
String	**Request ID**	
String	**Message**	Any unsent characters that should be sent before ending the chat.

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
15	**Invalid request ID**

Chat Chars Function

Send characters typed by the user.

Parameters		
String	**Session Id**	
String	**Request ID**	
String	**Message**	1 or more characters typed by the user.

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
15	**Invalid request ID**
16	**Function Not Allowed** (the dealer currently has control).

Chat Send Function

Hand control of the conversation to the dealer. Send any unsent characters that should be sent before handing control to the dealer.

Note: a style of Chat could be implemented that didn't send the characters as typed, but sent a whole line at once. This would be accomplished by calling **ChatSend()** alone and not calling **ChatChars()** at all.

Parameters		
String	**Session Id**	
String	**Request ID**	
String	**Message**	Any unsent characters that should be sent before handing control to the dealer.

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
15	**Invalid request ID**
16	**Function Not Allowed** (the dealer currently has control).

Chat Interrupt Function

Interrupt the dealer and take control of the Chat.

Parameters		
String	**Session Id**	
String	**Request ID**	
String	**Message**	Any unsent characters that should be sent before getting control from the dealer.

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
15	**Invalid request ID**
16	**Function Not Allowed** (the user currently has control anyway).

Chat Interrupt Done Function

Acknowledge dealer interrupt and transfer control to the dealer.

Note: it is not possible to decline a dealer interrupt; the purpose of this function is to allow any unsent characters to be sent to the dealer before the dealer can start typing.

Parameters		
String	**Session Id**	
String	**Request ID**	
String	**Message**	Any unsent characters that should be sent before handing control to the dealer.

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
15	**Invalid request ID**
16	**Function Not Allowed** (the dealer has not interrupted).

Accept Reverse Chat Property

Set to accept or reject "Reverse Chat" requests.

B2B trading programs using the API will typically set this to *off*, as they are not interested (or not able) to do Reverse Chat. On the other hand, programs implementing a trading GUI, should set the property to *on*. The property defaults to *off*.

Property		
Boolean	**Accept Reverse Chat**	Set to accept or reject "Reverse Chat" requests.

Chat Accepted Event

Reports success or failure of the **Chat** request.

Arguments		
String	**Session ID**	
String	**Request ID**	
Boolean	**Accepted**	
String	**Failure Message**	Empty unless login failed.

Failure Messages
"No dealer available at this time, please try again later"

Chat Chars Event

Characters from the dealer as they are typed.

Arguments		
String	**Session ID**	
String	**Request ID**	
String	**Message**	One or more characters typed by the dealer.

Chat Send Event

Message from dealer and transfer of control to the client.

Arguments		
String	**Session ID**	
String	**Request ID**	
String	**Message**	One or more characters typed by the dealer.

Chat Interrupt Done Event

Dealer has acknowledged client interrupt and the client now has control.

Arguments		
String	**Session ID**	
String	**Request ID**	

Chat End Event

The dealer has ended the Chat.

Arguments		
String	**Session ID**	
String	**Request ID**	
String	**Message**	Final message typed by the dealer.

Reverse Chat Request Event

The dealer has requested a Chat with the client.

A Chat session has been created, with the dealer in control.

Arguments		
String	**Session ID**	
String	**Dealer**	Dealer who wishes to Chat.
String	**Request ID**	Request ID of the dealer Chat.

Converted To Chat Event

The dealer has converted the current Deal to a Chat with the client.

The Deal has been terminated and a Chat session has been created, with the dealer in control. The Request ID of the Chat session is the same as that of the old Deal.

Arguments		
String	**Session ID**	
String	**Request ID**	Request ID of the Deal/Chat.
String	**Message**	Descripton of new Chat.

Split Trades

Summary of Split Trade Functions & Events

Functions	
Split Create	Create a (empty) split for a given market.
Split Clear	Empty a split.
Split Add	Add account to a pre-existing split.
Split Remove	Remove account from a pre-existing split..
Get Split	Get the details of a split.
Get Max Split Trade Size	Get the maximum trade size for a split account on a given market..
Get Min Split Trade Size	Get the minimum trade size for a split account on a given market..
Request Deal	Request a trade on a market *** **AMENDMENT** ***
Deal Response	Accept or reject a (re)quote *** **AMENDMENT** ***

Events	
Split List	Split details.

Flow of Split Trade Functions & Events

The definition of a split is created by calling **SplitCreate**. Splits are initially created empty and Account/Amount pairs are added to the split by calling **SplitAdd**. Account/Amount pairs can be removed by calling **SplitClear** or **SplitRemove**. A split definition can be inspected by calling **GetSplit**. This function generates a **SplitList** event for each Account/Amount pair.

The range of acceptable trade sizes for components of a split on a given market are returned by the **GetMax/MinSplitTrade Size** functions. In addition, the usual trade limits also apply to the whole bulk trade (as returned by the **GetMax/MinTradeSize** functions).

Bulk trades are done by passing an empty account into **RequestDeal** or **RequestQuote**. The API will then use the currently defined split for the market. An error will be generated if there is no split defined or if it is empty.

Split Create Function

Create a (empty) split for a given market.

If a split has already been defined for the market, the existing definition will be cleared.

Parameters		
String	**Session Id**	
String	**Market**	Market name
Integer	**Split Type**	**Not currently used – present for future compatibility.**

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.
String	**SplitId**	ID of the split, to be passed to the other split functions.

Errors	
22	**Not logged in**
2	**Invalid session ID**

Split Clear Function

Empty the split for a given market.

Parameters		
String	**Session Id**	
String	**SplitId**	

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
30	**No split defined for market**

Split Add Function

Add Account/Amount pair to a pre-existing split.

Parameters		
String	**Session Id**	
String	**SplitId**	
String	**Account**	

Parameters		
String	**Amount**	May include "Mill"

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
30	**No split defined for market**
25	**Invalid account**
7	**Invalid Amount**

Split Remove Function

Remove Account from a pre-existing split.

Parameters		
String	**Session Id**	
String	**SplitId**	
String	**Account**	

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.

Errors	
22	**Not logged in**
2	**Invalid session ID**
30	**No split defined for market**
25	**Invalid account**

Get Split Function

Get the details of the split on a given market.

Individual Account/Amount pairs are returned in **SplitList** events. An **EndOfList** event terminates the list.

Parameters		
String	**Session Id**	

Parameters		
String	**Market**	Market name

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.
String	**Request ID / Split ID**	

Errors	
22	**Not logged in**
2	**Invalid session ID**
30	**No split defined for market**

Get Max Split Trade Size Function

Gets the maximum trade size of any component of a split.

-1 is returned if there is no limit.

Parameters		
String	**Session Id**	
Integer	**Market No**	
Integer	**Trade Type**	0 – Spot, 1 – EFP, 2 - CFD
String	**Exchange**	Empty for Spot or CFD. "IMM" or "FINEX" for EFP trades.

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error
Integer	**Max Size**	

Errors	
2	**Invalid session id**
22	**Not logged in**
4	**Invalid market no**
5	**Invalid trade type**
6	**Invalid exchange**

Get Min Split Trade Size Function

Gets the minimum trade size of any component of a split.

-1 is returned if there is no limit.

Parameters		
String	**Session Id**	
Integer	**Market No**	
Integer	**Trade Type**	0 – Spot, 1 – EFP, 2 - CFD
String	**Exchange**	Empty for Spot or CFD. "IMM" or "FINEX" for EFP trades.

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error
Integer	**Min Size**	

Errors	
2	**Invalid session id**
22	**Not logged in**
4	**Invalid market no**
5	**Invalid trade type**
6	**Invalid exchange**

Request Deal Function ***AMENDMENT***

Request a trade on a market.
Supply the offered price in the **Price** field or leave blank to use the current price.

Leave the **Account** blank to use the current split.

Parameters		
String	**Session Id**	
Integer	**Market No**	
String	**Amount**	Digits and/or "mill", e.g. "100000" or "0.1 mill".
String	**Price**	Offered price or blank to use current price.
Integer	**Trade Type**	0 – Spot, 1 – EFP.
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades.
Integer	**Buy/Sell**	0 – Buy, 1 – Sell.
String	**Account**	Leave blank to specify a bulk trade.
String	**Client Ref**	Reference number supplied by client (optional)

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	Empty if error.

Errors	
2	**Invalid session id**
22	**Not logged in**
4	**Invalid market no**
7	**Invalid amount (there are min and max trade limits)**
5	**Invalid trade type**
6	**Invalid exchange**
19	**Invalid buy/sell**
8	**No current trade-able price for the market**
20	**No permission to trade this trade type**
21	**Market is closed or showing indicative prices**
25	**Invalid account**
26	**Price has invalid number of decimal places**
29	**Invalid Price**
30	**No split defined or split is empty**

Deal Response Function ***AMENDMENT***

Call to accept or reject a (re)quote. The trade will be automatically rejected if the timeout period has expired.

Parameters		
String	**Session Id**	
String	**Request Id**	
Integer	**Buy/Sell**	0 – Buy, 1 – Sell, 2 – NTG ("Nothing Done")
String	**Quote**	Optional confirmation of the deal price.

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error

Errors	
2	**Invalid session id**

Errors	
15	**Invalid request id**
19	**Invalid Buy/Sell/NTG**
9	**No trade / quote in progress**
10	**Timeout period of the quote has expired**
26	**Price has invalid number of decimal places**
29	**Invalid Price (price has not been offered)**

Split List Event

The details of an Account/Amount split pair.

Arguments		
String	**Session ID**	
String	**Request ID**	
String	**Market**	
String	**Account**	
String	**Amount**	
Integer	**Split Type**	**Not currently used – present for future compatibility.**

Split Orders

Summary of Split Order Functions, Properties & Events

Functions	
Get Order Split	Get the split for a particular order.
Create Order	Create a new order. *** **AMENDMENT** ***
Amend Order	Amend an existing working order. *** **AMENDMENT** ***
Cancel Order	Cancel an existing working order. *** **AMENDMENT** ***

Flow of Split Order Functions & Events

Split Orders work in a similar way to split trades. Splits are create by calling **SplitCreate** and modified by calling **SplitAdd**, **SplitRemove** and **SplitClear**. Passing an empty account into **CreateOrder** creates a bulk order and uses the split currently defined for the market.

The split on existing bulk orders can be retrieved by calling **GetOrderSplit**. This split can then be modified by calling **SplitAdd**, **SplitRemove** and **SplitClear** in the usual way. The modification is committed by calling **AmendOrder**.

Bulk orders are cancelled in the same way as normal orders by calling **CancelOrder**. **Note** that calling **CancelOrder** with the order number of a split will result in the whole bulk order being cancelled.

Get Order Split Function

Get the details of the split for a given bulk order.

Individual Account/Amount pairs are returned in **SplitList** events. An **EndOfList** event terminates the list.

This function is analogous to the **GetSplit** function and the split may be modified in a similar way by passing the return value of the function to **SplitAdd**, **SplitRemove** and **SplitClear**. **Note** that the actual order isn't modified until **AmendOrder** is called.

Parameters		
String	**Session Id**	
Integer	**Order No**	

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.
String	**Request ID / Split ID**	

Errors	
22	**Not logged in**
2	**Invalid session ID**
11	**Invalid Order Number or does not specify a bulk order**

Create Order Function *** AMENDMENT ***

Create a new order.

Parameters		
String	**Session Id**	
Integer	**Market No**	
String	**Amount**	
Integer	**Buy/Sell**	0 – Buy, 1 – Sell
Integer	**Stop/Limit**	0 – Limit, 1 – Stop
String	**Requested Price**	
Integer	**Trade Type**	0 – Spot, 1 – EFP

Parameters		
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades
Integer	**Good'Til**	0 - Cancelled, 1 - IMM Open, 2 - IMM Close, 3 - GMT
String	**Good'Til Time**	"ddmmyyyyhhmmss". Empty unless good'til "GMT"
String	**Instructions**	Special instructions, usually empty
String	**Account**	Leave blank to specify a bulk order.
String	**Client Ref**	Reference number supplied by client (optional)

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	Empty if error.

Errors	
2	**Invalid session id**
22	**Not logged in**
4	**Invalid market no**
7	**Invalid amount (there are min and max trade limits)**
19	**Invalid buy/sell**
23	**Invalid stop/limit**
24	**Invalid price (must be the correct side of the current market price)**
5	**Invalid trade type**
6	**Invalid exchange**
12	**Invalid Good'til**
13	**Invalid Good'til time (must be at least 5 minutes in the future)**
25	**Invalid account**
20	**No trade permission**
30	**No split defined or split is empty**

Amend Order Function * AMENDMENT *****

Amend an existing working order.

Note that if the order number of a split is provided (rather than the bulk), the amendment is applied to the whole bulk order.

Parameters		
String	**Session Id**	
Integer	**Order Number**	Of an already existing order.
String	**Amount**	New amount.
String	**Requested Price**	New requested price.
Integer	**Good'Til**	0 - Cancelled, 1 - IMM Open, 2 - IMM Close, 3 - GMT.
Time	**Good'Til Time**	Empty unless Good'til "GMT".
String	**Instructions**	New instructions.
String	**Account**	New account

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	Empty if error.

Errors	
2	**Invalid session id**
22	**Not logged in**
11	**Order number does not specify a valid working order**
7	**Invalid amount (there are min and max trade limits)**
24	**Invalid price (must be the correct side of the current market price)**
12	**Invalid Good'til**
13	**Invalid Good'til time (must be at least 5 min in the future)**
27	**No change to order was specified**
20	**No trade permission**
25	**Invalid account**
30	**No split defined or split is empty**

Cancel Order Function *** AMENDMENT ***

Cancel an existing working order.

Note that if the order number of a split is provided (rather than the bulk), the the whole bulk order is cancelled.

Parameters		
String	**Session Id**	

Parameters		
Integer	**Order Number**	Of an already existing order

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error
String	**Request Id**	Empty if error

Errors	
2	**Invalid session id**
22	**Not logged in**
11	**Order number does not specify a valid working order**

OCO/CGT Orders

Summary of OCO/CGT Orders Functions

Functions	
Create Complex Order	Create an OCO/CGT order.
Amend Complex Order	Amend an OCO/CGT order

Events	
Order List Event	Details of a particular order. *** Amendment ***

Flow of OCO/CGT Order Functions

OCO/CGT orders are created and amended with the functions **CreateComplexOrder** & **AmendComplexOrder**. These are analogous to **CreateOrder** & **AmendOrder**, but allow OCOs and CGTs to be created and amended. Note that split OCO/CGT orders can be created.

OCO/CGT orders are cancelled in the usual way by calling **CancelOrder**.

Create Complex Order Function

Create a new OCO/CGT order.

Parameters		
String	**Session Id**	
Integer	**Market No**	
String	**Amount**	
Integer	**Buy/Sell**	0 – Buy, 1 – Sell

Parameters		
Integer	**Stop/Limit**	0 – Limit, 1 – Stop
String	**Requested Price**	
Integer	**Trade Type**	0 – Spot, 1 – EFP
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades
Integer	**Good'Til**	0 - Cancelled, 1 - IMM Open, 2 - IMM Close, 3 - GMT
String	**Good'Til Time**	"ddmmyyyyhhmmss". Empty unless good'til "GMT"
String	**Instructions**	Special instructions, usually empty
String	**Account**	Leave blank to specify a bulk order.
String	**Client Ref**	Reference number supplied by client (optional)
String	**Cgt Price**	Price of the contingent to the primary order. Leave blank to mean no CGT (and hence no CGT OCO).
String	**Cgt Stop Limit**	Stop/Limit of the contingent.
String	**Cgt Oco Price**	Price of the OCO to the contingent order. Leave blank to mean no CGT OCO.
String	**Oco Price**	Price of the OCO to the primary order. Leave blank to mean no OCO (and hence no OCO CGT or OCO CGT OCO).
String	**Oco Cgt Price**	Price of the contingent to the primary OCO order. Leave blank to mean no OCO CGT (and hence no OCO CGT OCO).
String	**Oco Cgt Stop Limit**	Stop/Limit of the contingent. to the primary OCO order.
String	**Oco Cgt Oco Price**	Price of the OCO to the contingent of the primary OCO order. Leave blank to mean no OCO CGT OCO.

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	Empty if error.

Errors	
2	**Invalid session id**
22	**Not logged in**

Errors	
4	**Invalid market no**
7	**Invalid amount (there are min and max trade limits)**
19	**Invalid buy/sell**
23	**Invalid stop/limit**
24	**Invalid price (must be the correct side of the current market price)**
5	**Invalid trade type**
6	**Invalid exchange**
12	**Invalid Good'til**
13	**Invalid Good'til time (must be at least 5 minutes in the future)**
25	**Invalid account**
20	**No trade permission**
30	**No split defined or split is empty**

Amend Complex Order Function

Amend an existing complex working order.

Note that if the order number of a split is provided (rather than the bulk), the amendment is applied to the whole bulk order.

Parameters		
String	**Session Id**	
Integer	**Order Number**	Of an already existing order.
String	**Amount**	New amount.
String	**Requested Price**	New requested price.
Integer	**Good'Til**	0 - Cancelled, 1 - IMM Open, 2 - IMM Close, 3 - GMT.
Time	**Good'Til Time**	Empty unless Good'til "GMT".
String	**Instructions**	New instructions.
String	**Account**	New account.
String	**Cgt Price**	Price of the contingent to the primary order. Leave blank to mean no CGT (and hence no CGT OCO).
String	**Cgt Stop Limit**	Stop/Limit of the contingent.
String	**Cgt Oco Price**	Price of the OCO to the contingent order. Leave blank to mean no CGT OCO.

Parameters		
String	**Oco Price**	Price of the OCO to the primary order. Leave blank to mean no OCO (and hence no OCO CGT or OCO CGT OCO).
String	**Oco Cgt Price**	Price of the contingent to the primary OCO order. Leave blank to mean no OCO CGT (and hence no OCO CGT OCO).
String	**Oco Cgt Stop Limit**	Stop/Limit of the contingent. to the primary OCO order.
String	**Oco Cgt Oco Price**	Price of the OCO to the contingent of the primary OCO order. Leave blank to mean no OCO CGT OCO.

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	Empty if error.

Errors	
2	**Invalid session id**
22	**Not logged in**
11	**Order number does not specify a valid working order**
7	**Invalid amount (there are min and max trade limits)**
24	**Invalid price (must be the correct side of the current market price)**
12	**Invalid Good'til**
13	**Invalid Good'til time (must be at least 5 min in the future)**
27	**No change to order was specified**
20	**No trade permission**
25	**Invalid account**
30	**No split defined or split is empty**

Order List Event ***** AMENDMENT *****

Describes a single order within the order history:

Arguments		
String	**Session Id**	Session that generated the event.
String	**Request Id**	Request that generated the event.
Integer	**Order Number**	

Arguments		
Integer	**Parent**	For split orders, contains the order number of the bulk order, otherwise 0.
String	**Amount**	
String	**Market**	
Integer	**Buy/Sell**	0 – Buy, 1 – Sell.
Integer	**Stop/Limit**	0 – Limit, 1 – Stop.
String	**Requested Price**	
String	**Current Price**	
Integer	**Trade Type**	0 – Spot, 1 – EFP.
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades.
Integer	**Good'Til**	0 - Cancelled, 1 - IMM Open, 2 - IMM Close, 3 - GMT
String	**Good'Til Time**	Empty unless Good'til time is set to "GMT"
String	**Instructions**	Special instructions, usually empty
String	**Creation Time**	Time order was created
String	**Amendment Time**	Time order was last amended
String	**Execution Time**	Time order was executed or cancelled, empty if not
String	**Dealer Name**	Dealer who executed or cancelled order
String	**Account**	Empty for bulk orders
String	**Client Ref**	Reference number supplied by client (optional)
Integer	**Primary No**	For CGT's, contains the order number of the primary, otherwise 0.
Integer	**OCO With No**	Contains the order number of any OCO to this order, 0 if none.
Integer	**Contingent No**	Contains the order number of the CGT order, 0 if none.

Prices *** AMENDMENT ***

Summary of Prices Functions & Events

Events	
Price Change	A price has changed *** AMENDMENT ***

Price Change Event

A price has changed.

Arguments		
String	**Session Id**	Session that generated the event.
String	**Request Id**	Request that generated the event.
Integer	**Market No**	Unique market identifier (small positive integer).
String	**Market**	Market name (may not be unique across pages).
String	**Bid**	Bid price.
Char	**Bid direction**	'S' - steady, 'U' - up or 'D' – down.
String	**Bid Liquidity**	Largest amount available at this bid price (empty if no information available)
String	**Ask**	Ask/Offer price.
Char	**Ask direction**	'S' - steady, 'U' - up or 'D' – down.
String	**Ask Liquidity**	Largest amount available at this offer price (empty if no information available)
String	**High**	Day's highest Bid.
String	**Low**	Day's lowest Offer.
Integer	**Market state**	0 - Closed, 1 - Open, 2 - Indicative price only, 3 - "Request For Quote" Only, 4 - "Telephone Only"
String	**Timestamp**	Time of change in GMT.

Miscellaneous

Summary of Miscellaneous Properties

Properties	
App Name	The (short) name of the application using the API.
Version No	The version number of the application using the API.
Request Liquidity	Liquidity information will be generated (if available), in addition to normal price change data.

Summary of Miscellaneous Functions

Functions	
Get Client Permissions	Get the trading permissions for the client.
Login	Start a new session *** **AMENDMENT** ***

Functions	
List Markets	List all the markets on a given page.
Get Market Details	Get the details of a particular market.

Events	
Client Permissions	Returns the client's permissions.
Market List	Returns the details of a market.
Server Response Time	Provides an indication of the current connection quality to the server.
Liquidity	Returns liquidity information.

Get Client Permissions Function

Get the trading permissions for the client.

Parameters		
String	**Session Id**	

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error
String	**Request ID**	

Errors	
2	**Invalid session id**
22	**Not logged in**

Client Permissions Event

Returns the client's permissions.

This event is fired after calling the **GetClientPermissions** function and thereafter whenever the permissions change.

The low-word of the permissions contains the trade-type permissions of the client (ie what they are allowed to trade: Spot, EFP etc). This is an amalgamation of the permissions on the client's individual accounts.

The high-word contains permissions relating to how they trade (eg RFQ permission).

Arguments		
String	**Session ID**	
String	**Request ID**	
Integer	**Permissions**	Bit flags: 0x0001 – Spot. 0x0002 – EFP 0x0004 – CFD 0x10000 – "Request for Quote" Allowed. 0x20000 – "Click & Trade" Allowed. 0x40000 – No Bullion

Login Function * AMENDMENT *****

Initiate a new session with the server. Progress and confirmation of the login is done via the **Login** event.

Parameters		
String	**User ID**	
String	**User Name**	
String	**Password**	
Integer	**Address**	0 means connect to the default (ie main) server. Anything else means connect to the alternate server. In current configurations, this would be the "demo" server. **Note**: In the future, more than one alternate address may be provided.
Integer	**Language**	The Microsoft language ID to be used for server messages. Typical values are: 0x000 – Current user's default language. 0x009 – English (Generic). 0x404 – Traditional Chinese **Note**: API generated status messages and error text will always be in English.

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Session Id**	Empty if error.

Errors	
18	**No login details**
1	**Already logged in**

Login Event *** AMENDMENT ***

Reports success or failure of the **Login** request.

Arguments		
String	**Session Id**	
Boolean	**Accepted**	
String	**Failure Message**	Empty unless login failed.
Integer	**Failure Code**	Not relevant unless login failed.

Failure Code	
0	Unable to connect to deal server (retry in progress)
1	Encrypted session is mandated by the server
2	API interface software is out-of-date
3	Login details are incorrect
4	Account has been disabled
5	The dealing service is temporarily unavailable
6	Account has expired (demo accounts only)

List Markets Function

List all the markets and their details on a given price page. Details are returned by **Market List** events (see below).

Parameters		
String	**Session Id**	
Integer	**Page Number**	

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
3	**Invalid Page Number**

Get Market Detail Function

Get the details of a particular market. Details are returned by **Market List** events (see below).

Parameters		
String	**Session Id**	
Integer	**Market No**	Market number

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
4	**Invalid Market Number**

Market List Event

Returns the details of a market.

Arguments		
String	**Session Id**	
String	**Request Id**	
Integer	**Market No**	
String	**Market**	Market name
Integer	**Dec Places**	Number of decimal places
String	**Exchange**	Exchange name (currently blank unless EFP)
Integer	**Trade Type**	0 – SPOT, 1 – EFP, 2 - CFD

App Name Property

The (short) name of the application using the API.

The application name uniquely identifies the application at the server. The application name and version number are used for logging and version checking by the server.

The name is limited to 10 characters and cannot contain full stops ('.').

Property		
String(10)	**App Name**	The (short) name of the application using the API.

Version No Property

The version number of the application using the API.

The version number is an integer representation of the conventional "x.xx" version number string multiplied by 100, eg version "1.00" is represented as 100.

The application name and version number are used for logging and version checking by the server. Note that the version number is checked independently of the underlying API version number (which cannot be changed).

Property		
String	**Version No**	The version number of the application using the API.

Server Response Time Event

Provides an indication of the current connection quality to the server.

The connection quality is divided into three bands: Green (good), Amber and Red (poor). The timing definitions of the bands are defined by server properties.

Arguments		
String	**Session Id**	Session that generated the event
String	**Time**	0 – Green band, 1 – Amber, 2 – Red.

Request Liquidity property

Liquidity information (if available) will be generated by the API, in addition to normal price changes, when calls to **RequestPrices** are made. The information will be returned via the new Liquidity event (see below). An event which will be generated for each change in the market liquidity.

Property		
Boolean	**Request Liquidity**	If TRUE, liquidity information will be requested from the server (default is FALSE).

Liquidity Event

Provides latest liquidity information.

Arguments		
String	**Session Id**	Session that generated the event

Argum en ts		
String	**Request Id**	Request Id of the **RequestPrices** call that generated this event.
Integer	**Market No**	Market number.
String	**Bid**	Bid price.
String	**Bid Liquidity**	Largest amount available at this bid price.
String	**Ask**	Ask/Offer price.
String	**Ask Liquidity**	Largest amount available at this offer price.
String	**Timestamp**	Time of change (in GMT).

Positions Functions and Events

Summary of Positions Functions

Func tions	
Account Summary	Get an account summary.
Open Positions	Get current open positions for an account.
Get Position Detail	Get the details of a particular position.
Position Deals	Get the deals contributing to a given position.
Get Position Deal Detail	Get the details of a particular deal.

Summary of Positions Events

Event s	
Account Summary Event	Returns account summary details.
Position Event	Returns position information.
Position Deal Event	Returns (position) deal information.

AccountSummary Function

Call to get the financial summary information for an account.

Information is returned by a single **AccountSummary** event. This event is generated continuously, every time the information changes (there is no need to keep requesting it).

Parameters		
String	**Session Id**	
String	**Account**	Account

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
25	**Invalid account**

OpenPositions Function

Get open positions information for an account.

Position information is returned by zero or more **Position** events. Position changes are notified in the usual way by **ItemChanged** events.

Parameters		
String	**Session Id**	
String	**Account**	Account

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
25	**Invalid account**

GetPositionDetail Function

Gets the details of a particular open position.

Position information is returned by a **Position** event.

Parameters		
String	**Session Id**	
String	**Reference**	Position reference

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
31	**Invalid position reference**

PositionDeals Function

Returns the deals that have contributed to a position.

Position information is returned by zero or more **PositionDeal** events. Changes are notified in the usual way by **ItemChanged** events.

Parameters		
String	**Session Id**	
String	**Reference**	Position reference

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
31	**Invalid position reference**

GetPositionDealDetail Function

Gets the details of a particular (position) deal.

Parameters		
String	**Session Id**	

Parameters		
String	**Reference**	Deal reference

Returns		
Integer	**Error**	0 if successful, otherwise error code.
String	**Error Message**	Empty unless error.
String	**Request Id**	

Errors	
2	**Invalid session id**
22	**Not logged in**
28	**Invalid deal reference**

Account Summary Event

Returns account summary information.

All monetary fields in this event are amounts in the client's base currency.

Arguments		
String	**Session Id**	
String	**Request Id**	
String	**Cash Equity**	Cash equity amount.
String	**Open Trade Equity**	Open trade equity (*aka* Unrealised P&L) amount.
String	**Margin**	Total Margin requirement (including open orders, if applicable)
String	**Order Margin**	Margin requirement of open orders. **Note:** this field will be zero except on systems that have margining of open orders enabled.
String	**Net Equity**	Cash equity + Open Trade equity
String	**Credit Allocated**	Credit allocated. Positive amount if credit available. Usually this will be 0, but could also be less than 0.
String	**Trading Resources**	Total amount available for trading: Net Equity – Margin Requirement + Credit Allocated.
String	**Base Currency**	Account's base currency: 3 character ISO currency code.

Position Event

Returns open position information.

Arguments		
String	**Session Id**	
String	**Request Id**	
String	**Reference**	Position reference
String	**Market**	Market name.
String	**ValueDate**	Value (settlement) date in "dd/mm/yyyy" format. This field will be empty for markets that don't have a value date.
Integer	**Position**	Signed position amount. Positive means long and negative means short.
String	**AveragePrice**	Average price of the position.
String	**CurrentPrice**	Current price to **close** the position.
String	**UnrealisedPandL**	Unrealised P&L at current price (in currency 2).
String	**RealisedPandL**	Realised P&L (in currency 2).
String	**Margin**	Margin requirement at current price (in currency 1).
String	**PandLCurrency**	Currency of the P&L amounts (ie currency 2). 3 character ISO currency code.
String	**BaseCurrency**	Account's base currency: 3 character ISO currency code.

Position Deal Event

Returns information about a deal contributing to a position.

Arguments		
String	**Session Id**	
String	**Request Id**	
String	**Reference**	Deal reference
String	**Amount**	Deal amount.
String	**Price**	Deal price.
Integer	**BuySell**	0 = BUY, 1 = SELL.
String	**Timestamp**	Time and date of deal.
Integer	**State**	0 = Open, 1 = Part-closed, 2 = Closed.
String	**Margin**	Margin requirement at current price (in currency 1).
String	**UnrealisedPandL**	Unrealised P&L at current price (in currency 2).

Arguments		
String	**RealisedPandL**	Realised P&L (in currency 2).