Ariel Server API Specification

Author: *Mike Woolley*
Date: *6th Sept 2004*
Document Version: *2.7*
API Version: *2.10*

Introduction

This document presents the API for third parties to do trades and orders with the Ariel Deal Server directly, without using the Ariel Dealing Client program.

The current implementation offers a subset of existing Deal Server functionality (e.g. OCO/Contingent orders are not in the specification), although it is envisaged that the API will be enhanced in future phases to include these other features.

This document is in two parts. The first part presents the functions available and the values passed to and received from the server. The second part describes the interface currently implemented, which is via an ActiveX control. It is envisaged that different interface methods will be added in the future.

Implementation notes are shown in the text [*like this*].

Contents

Functions

The basic structure of the API is an asynchronous, event-driven architecture. Everything takes place in the context of a "session", which represents the activity of one dealing system user. Requests are made on the session, such as to get prices or make a trade and all progress and results of requests are returned as asynchronous events. Each event is tagged with the ID of the session and request that generated the event, allowing requests and responses to be tied together.

This section presents the trading functions available and lists the input and output parameters, events generated and possible errors raised. All the errors are summarised in a table at the end of the document (page 24).

Functions and events are split into the following categories:

1. Session Management.
2. Prices.
3. Trades.
4. Orders.
5. General Functions and Events.

Session Management

- [*In the current implementation, only one session can be created at once. This means that only one Ariel user can be logged in at once to each instance of the API. However, that user can do multiple and concurrent trades and orders on any of the accounts managed by that user.*]
- [*It is envisaged that multiple concurrent sessions in a single instance of the API will be supported in the future.*]

Summary of Session Management Functions & Events

Functions	
Login	Start a new session
Logout	Terminate a session
List Accounts	Get list of user's accounts

Events	
Login	Login success or failure
Account List	Information on an account

Login Function

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

Parameters	
String	**User ID**
String	**User Name**
String	**Password**

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**

Errors	
1	**Already logged in**

Login Event

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

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

Failure Messages
"Login details are incorrect"
"Account has been disabled"
"Account has expired" (demo accounts only).
"API interface software is out-of-date"

Logout Function

Terminate a given session.

Note that **logout** is not allowed if a deal or other critical process is in progress (error **16** will be returned).

Parameters	
String	**Session Id**

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

Errors	
22	**Not logged in**
2	**Invalid session Id**
16	**Abort not allowed**

List Accounts Function

Get a list of the user's accounts. Account information is returned in a sequence of **Account** events. The list is terminated by receipt of the **End of List** Event.

If the account list changes, the **List Changed** event will be sent on this Request Id. The new list can be retrieved by calling **List Accounts** again.

Parameters	
String	**Session Id**

Returns		
Integer	**Error**	
String	**Error Message**	Empty unless error.
String	**Request Id**	Empty if error.

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

Account Event

Information about an individual account.

Arguments		
String	**Session Id**	Session that generated the event
String	**Request Id**	Request that generated the event
String	**Name**	Account name
Integer	**Trade Permissions**	Trade types allowed on account. Bit flags: bit 0 – SPOT, bit 1 – EFP, bit 2 – CFD.

Prices

A continuous feed of prices is available. Prices are split into pages by trade type/exchange.

Summary of Prices Functions & Events

Functions	
Request Prices	Request a continuous feed of prices

Events	
Price Change	A price has changed

Request Prices Function

Request a continuous feed of prices. Prices are returned by the **Price Change** event, described below.

Parameters		
String	**Session Id**	
Integer	**Page number**	0 = Spot, 1=CFD, 2 = EFP FINEX, 3 = EFP IMM. ***Note:*** *All these pages may not be available on the system. Please consult your vendor.*

Returns	
Integer	**Error**
String	**Error Message**
String	**Request Id**

Errors	
22	**Not logged in**
2	**Invalid session id**
3	**Invalid page no**

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	**Ask**	Ask/Offer price.
Char	**Ask direction**	'S' - steady, 'U' - up or 'D' – down.
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.

Trades

There are two ways to do a deal. The **Request Deal** function allows an immediate buy or sell at the current advertised price (*aka* "Click & Trade"). The **Request Quote**

function requests a "two-way" (e.g. 0.9999-02) price provided by a dealer and the user can then decide whether to buy or sell.

[*Bulk trades (aka splits) are not currently supported by the API interface*]

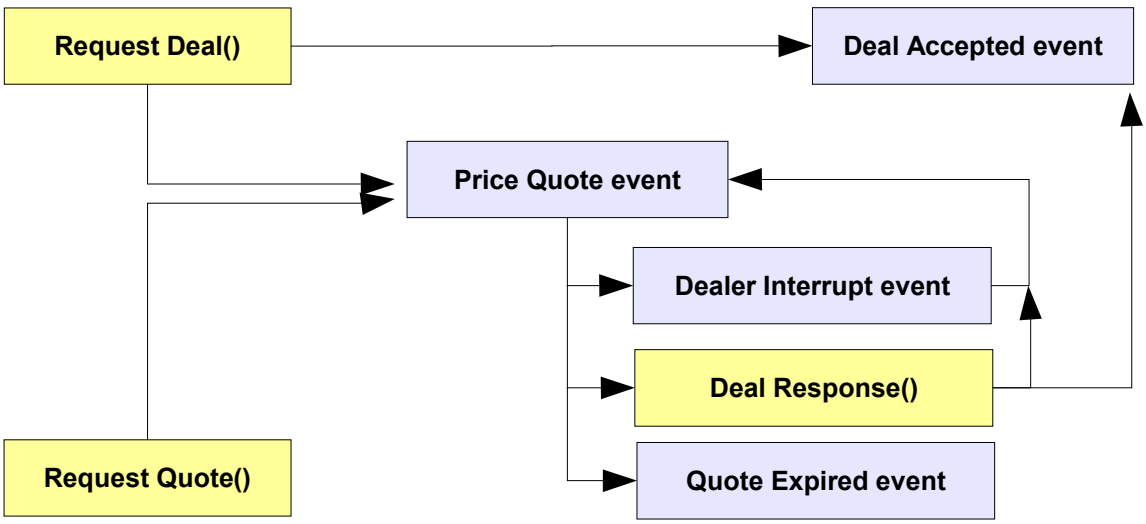
[*Forwards are also not currently supported by the API interface*]

Summary of Trade Functions & Events

Functions	
Request Deal	Request a trade on a market at the current advertised price
Request Quote	Request a quote for a trade on a market.
Deal Response	Accept or reject a (re)quote.
Trade History	Get a list of recent trades.
Get Trade Detail	Gets the details of a specific trade.

Events	
Deal Accepted	A deal has been accepted or rejected by the dealer.
Price Quote	Deal has been (re)quoted by the dealer.
Dealer Interrupt	The dealer has interrupted and is providing a new quote
Quote Expired	The current quote has expired
Trade History	Describes a single trade in the trade history

Flow of Trade Functions & Events



Request Deal Function

Request a trade on a market at the current advertised price. Progress and confirmation of the trade is done by the Trade events described below.

Parameters		
String	**Session Id**	
Integer	**Market No**	
String	**Amount**	Digits and/or "mill", e.g. "100000" or "0.1 mill".
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**	
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**

Deal Accepted Event

Deal has been accepted or rejected.

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

Arguments		
String	**Request Id**	Request that generated the event
Boolean	**Accepted**	Deal was accepted or rejected
String	**Dealer message**	Confirmation or rejection message from dealer
String	**Accept Time**	Time deal was done (GMT)
Integer	**Deal Number**	0 if deal not accepted
String	**Price**	Empty if deal not accepted
String	**Cash Price**	Empty unless an EFP deal has been accepted

Price Quote Event

Deal has been (re)quoted. The **Deal Response** function must be called within the timeout period to Buy/Sell at the new price.

Arguments		
String	**Session Id**	Session that generated the event
String	**Request Id**	Request that generated the event
String	**Price**	Price quote for trade This will be a two-way price if a quote was requested
Integer	**Timeout**	Time in secs that the new price is valid for (usually 5)

Dealer Interrupt Event

Whilst waiting for a Deal Response, the dealer has interrupted and will either provide a new quote or terminate the deal (via further events).

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

Deal Response Function

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")

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

Errors	
2	**Invalid session id**
15	**Invalid request id**
19	**Invalid Buy/Sell/NTG**
9	**No trade / quote in progress**
10	**Timeout period of the quote has expired**

Request Quote Function

Request a quote for a trade on a market. Progress and confirmation of the trade is done by the Deal events described above. Acceptance of the quote is done by calling the **Deal Response** function.

Parameters		
String	**Session Id**	
Integer	**Market No**	
String	**Amount**	Digits and "mill", e.g. "100000" or "0.1 mill" .
Integer	**Trade Type**	0 – Spot, 1 – EFP.
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades.
String	**Account**	
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**

Errors	
6	**Invalid exchange**
25	**Invalid account**

Quote Expired Event

The current quote (timeout) has expired.

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

Trade History Function

Lists the last 14 days deals and executed orders. Trades are returned via **Trade History** events. The **End of List** event signals that all trades in the list have been received.

If the trade list changes, the **List Changed** event will be sent on this Request Id. The new list can be retrieved by calling **Trade History** again.

Parameters	
String	**Session Id**

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.

Trade History Event

Describes a single trade in the trade history.

Arguments		
String	**Session Id**	Session that generated the event
String	**Request Id**	Request that generated the event
String	**Deal Number**	Eg 801168, 801169O, 801170T etc
String	**Market**	
String	**Amount**	

Arguments		
String	**Price**	
Integer	**Buy/Sell**	0 – Buy, 1 – Sell
Integer	**Trade Type**	0 – Spot, 1 – EFP
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades
String	**Cash Price**	Empty unless an EFP trade
String	**Dealer Name**	
String	**Time**	Time the deal was done (GMT)
String	**Account**	Empty for bulk trades
String	**Client Ref**	Reference number supplied by client

Get Trade Detail Function

Gets the details of a specific trades by deal reference.

The details are returned in a single Trade History event.

Parameters		
String	**Session Id**	
String	**Deal Number**	Deal Reference

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**
28	**Invalid deal number**

Orders

Orders can be created, amended and cancelled, subject to confirmation by a dealer.

Current working orders can be listed, as well as orders executed and cancelled in the last 14 days.

Summary of Order Functions & Events

Functions	
Create Order	Create a new order
Amend Order	Amend an existing working order
Cancel Order	Cancel an existing working order
List Working Orders	Get a list of the current working orders
List Cancelled Orders	Get a list of recent cancelled orders
List Executed Orders	Get a list of recent executed orders
Get Order Detail	Get details of a specific order
Request Order Fill Notify	Requested notification of order execution

Events	
Order Accepted	A order has been accepted or rejected by the dealer
Order List	Information about a single order

[*OCO/Contingent orders are not currently supported by the API interface, but it is envisaged that they will be in a future release*]

[*Bulk orders (aka splits) are also not currently supported by the API interface*]

Create Order Function

Create a new order. Progress and confirmation of the New Order request is done by the Order events described below.

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
String	**Exchange**	Empty for Spot, "IMM" or "FINEX" for EFP trades

Parameters		
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**	
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**

Order Acceptance Event

An order has been accepted or rejected by a dealer.

Arguments		
String	**Session Id**	Session that generated the event.
String	**Request Id**	Request that generated the event.
Boolean	**Accepted**	Order has been accepted by a dealer.
String	**Dealer Message**	Usually empty, unless dealer rejected the order request.

Arguments		
Integer	**Order Number**	0 if order request not accepted.

Amend Order Function

Amend an existing working order. Progress and confirmation of the Amend Order request is done by the Order Acceptance event previously described.

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.

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**

Cancel Order Function

Cancel an existing working order. Progress and confirmation of the Cancel Order request is done by the Order events previously described.

Parameters		
String	**Session Id**	
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**

List Working Orders Function

Lists current working orders. Orders are returned via Order List events described below.

If the working order list changes, the **List Changed** event will be sent on this Request Id. The new list can be retrieved by calling **List Working Orders** again.

Parameters	
String	**Session Id**

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**

Order List Event

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.

Arguments		
Integer	**Order Number**	
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)

List Cancelled Orders Function

Lists orders cancelled in the last 14 days. Orders are returned via Order List events described previously.

If the cancelled order list changes, the **List Changed** event will be sent on this Request Id. The new list can be retrieved by calling **List Cancelled Orders** again.

Parameters	
String	**Session Id**

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**

List Executed Orders Function

Lists orders filled in the last 14 days. Orders are returned via Order List events described previously.

If the executed order list changes, the **List Changed** event will be sent on this Request Id. The new list can be retrieved by calling **List Executed Orders** again.

Parameters	
String	**Session Id**

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**

Get Order Detail Function

Gets the details of a specific order by order number.

The details are returned in a single **Order List** event.

Parameters		
String	**Session Id**	
Integer	**Order Number**	Of an already existing working 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**

Errors	
11	**Order number does not specify a valid working order**

Request Order Fill Notify

Requests notification of order execution.

When an order is filled a single **Order List** event is sent, giving the details of the executed order.

Parameters	
String	**Session Id**

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**

General

General functions and events which can be called or generated in multiple contexts.

Summary of General Functions & Events

Functions	
Get Minimum Trade Size	Get the minimum size for a trade or order
Get Maximum Trade Size	Get the maximum size for a trade or order
Abort Request	Cancels a request
Get Version	Get the version number of the API
Get Server Address	Get the host name or IP address of the server

Events	
Connected To Server	A successful connection to the server has been made
Dealer Allocated	A dealer has been allocated to deal with the request

Events	
Lost Connection	The connection to the server has been lost
Dealer Held	The dealer has put a request on hold
Status Message	Provides a user-oriented status message
End of List	Indicates that all the items in the list have been retrieved
List Changed	A list has changed
Item Changed	A list item has changed
Server Time	Provides an indication of the current server time

Get Minimum Trade Size Function

Get the minimum size for a trade or order.

-1 is returned if there is no limit.

Parameters		
String	**Session Id**	
Integer	**Market No**	
Integer	**Trade Type**	0 – Spot, 1 – EFP.
String	**Exchange**	Empty for Spot, "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**

Get Maximum Trade Amount Function

Return the maximum size for a trade or order.

-1 is returned if there is no limit.

Parameters		
String	**Session Id**	
Integer	**Market No**	
Integer	**Trade Type**	0 – Spot, 1 – EFP.
String	**Exchange**	Empty for Spot, "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**

Abort Request Function

Abort or cancel a request.

Note that not all requests can be aborted. In particular, deals cannot be cancelled when they've reached the final stages.

Parameters		
String	**Session Id**	
String	**Request Id**	

Errors	
2	**Invalid session id**
15	**Invalid request id**
22	**Not logged in**
16	**Abort not allowed**

Get Version Function

Get the version of the API.

Returns		
String	**Version**	

Get Server Address Function

Get the host name or IP address of the server.

Parameters		
String	**Session Id**	

Returns		
String	**Address**	The host name or IP address of the server

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

Connected to Server Event

A successful connection to the server has been made.

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

Dealer Allocated

A dealer has been allocated to deal with the request.

Arguments		
String	**Session Id**	Session that generated the event.
String	**Request Id**	Request that generated the event.
String	**Dealer name**	

Lost Connection Event

The connection to the server has been lost.

Note that all outstanding requests that haven't been committed by the server will have been cancelled. Depending on where in the process a request is, it may be necessary to list trades and orders upon reconnection to confirm whether a request has been done or not.

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

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

Status Message Event

Provides a user-orientated message describing a change in the current state suitable for displaying on a status bar or for debugging (eg "Connected to server") .

Arguments		
String	**Session Id**	Session that generated the event
String	**Request Id**	Request that generated the event
String	**Message**	Status message

Dealer Held Event

The dealer has put this request on hold. The same or another dealer will pick it up again shortly.

Arguments		
String	**Session Id**	Session that generated the event
String	**Request Id**	Request that generated the event
String	**Message**	Hold message

End of List Event

There are no more items in the list being retrieved.

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

List Changed Event

The list corresponding to the supplied **Request Id** has changed.

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

Item Changed Event

The item in the list corresponding to the supplied **Request Id** has changed. The item is identified by it's reference (deal reference number or order number).

Arguments		
String	**Session Id**	Session that generated the event
String	**Request Id**	Request that generated the event.
String	**Reference**	Item reference (deal reference or order number)
Integer	**Change Type**	0 = Item Added, 1 = Item Amended, 2 = Item Deleted.

Server Time Event

Provides an indication of the current server time.

Arguments		
String	**Session Id**	Session that generated the event
String	**Time**	Server time to the nearest second

Table of Error Codes

Number	Error
0	Initialisation failure (eg couldn't find configuration file "API.PPC")
1	Already logged in
2	Invalid session Id
3	Invalid page number
4	Invalid market number
5	Invalid trade type
6	Invalid exchange
7	Invalid amount
8	No current trade-able price for market
9	No quote in progress
10	Quote expired
11	Invalid order number
12	Invalid good'til
13	Invalid good'til time
14	Internal error (shouldn't occur)
15	Invalid request id
16	Abort not allowed
17	Not implemented
18	No login details
19	Invalid buy/sell

Number	Error
20	No permission for trade
21	Market not open
22	Not logged in
23	Invalid stop/limit
24	Price is too close to the market
25	Invalid account
26	Price has an invalid number of decimal places for the market
27	No changes specified for an amendment
28	Invalid deal number

Interface

As stated in the Introduction, there are various ways the trading functions could be exposed to third-parties, but the current implementation is as an ActiveX control, which is described below.

ActiveX control

Ariel provides an ActiveX control (OCX), which third-party programs will use to invoke trading functions and receive trading events. The ActiveX control itself uses Ariel's trading protocol behind the scenes to talk to the Dealing System. By default, this uses port 1000 to talk to the Dealing System and this port will have to be opened up on any intervening firewalls.

Changes to the API specification

- Booleans are implemented as short integers.
- Errors are reported using the COM exception handler rather than returned by reference parameters. Error codes are offset by 1000 from those shown in the specification, eg "Already logged in" is error number 1001.
- Request functions return the request id as a string (BSTR).