Refinitiv REDI EMS

API Specification

REFINITIV™

Document Version 2.4
Date of issue: 3 December 2019

API Disclaimer

REDI Global Technologies LLC (member FINRA), REDI Technologies Limited (FCA #612490) and their affiliates (collectively, "REDI") may make available application programming interface ("API") specifications, including sample code, for use solely in connection with a user's internal non-commercial development purposes, in accordance with a user's agreement with REDI and this REDI EMS API Specification document. For the avoidance of doubt, a user must enter into an agreement with REDI in order to be able to connect to the REDI EMS using this API specification document. No right, title or interest is granted in or to the API specifications and user agrees at all times to treat these materials in a confidential and secure manner. By utilizing API specifications provided by REDI, the user of the API specifications understands that use of such specifications may affect live trading. The user of the API specifications assumes all responsibility for any action, or lack thereof, taken in a live trading environment including but not limited to execution of orders, cancellation of orders, modification of orders, analysis of market data, or the calculation of any position, order, complex order, or spread. The API specifications are provided on an "as is" basis and without warranty of any kind. REDI specifically disclaims all warranties for use of API specifications, express or implied, including implied warranties of merchantability, fitness for a particular purpose, title, infringement, and operation of the API specifications with the REDI EMS. REDI does not warrant that use of the API specifications will be uninterrupted, error-free, timely, complete, accurate or that defects in the foregoing can or will be corrected. REDI does not make any warranties as to the results to be obtained from use of the API specifications and use of the API specifications and any reliance thereon is at the user's sole risk. No oral or written information or advice given by REDI shall create any warranties or in any way increase the scope of REDI's obligations. This document has been prepared for informational purposes only. REDI is not providing any services or advice by virtue of providing this document. The information in this document is subject to change without notice. Except as otherwise expressly permitted herein, the modification, reformatting, copying, downloading, storing, reproduction, or retransmission of API specifications is prohibited. All materials and services provided to you by REDI are governed by the terms of any applicable agreements with REDI. IN NO EVENT WILL REDI OR ITS THIRD PARTY PROVIDERS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION DIRECT OR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSSES OR EXPENSES ARISING IN CONNECTION WITH API SPECIFICATIONS.

This document is intended for only institutional accounts (as defined by FINRA) and Eligible Counterparties (as defined by the FCA), and it is not intended for retail investors/persons. REDI services and related services are not available in all jurisdictions.

Contents

Chapter 1 Introduction

1.1 Purpose

This document provides a general guideline for the development of applications vis-à-vis the Refinitiv REDI Execution Management System (referred to throughout as the REDI EMS).

1.2 Audience

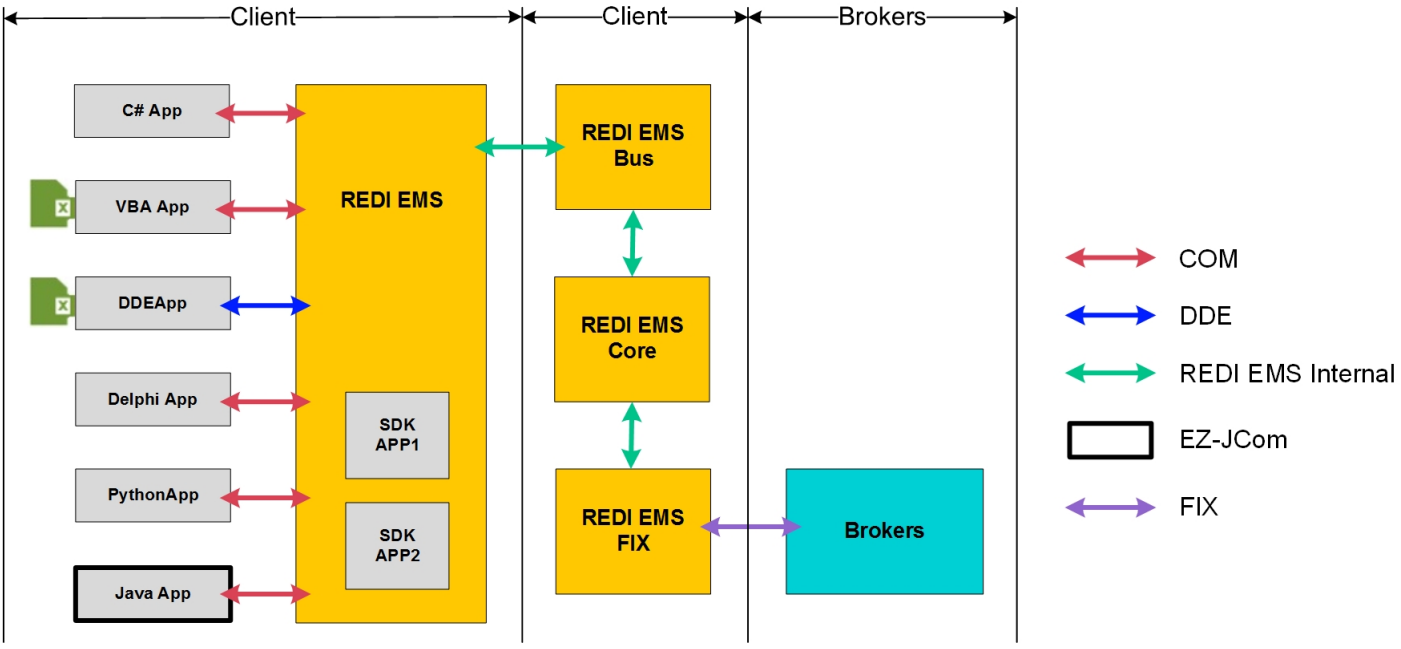
The intended audiences of this document are software developers/ engineers interested in or are currently leveraging the REDI EMS API to build custom applications.

1.3 Overview

The REDI EMS offers Application Programming Interface (hereafter “API” or “APIs”) tools that empower software developers/ engineers to build custom applications to interact with and maximize the potential of our flagship REDI EMS.

It is possible, for example, for an event-driven trader to leverage a custom application built via the REDI EMS API to trigger trades based on a combination of REDI EMS provided market data and pre-defined ‘trading opportunities’ (say, a minimum spread between bid and ask prices for a list of symbols) and much more.

The REDI EMS’s API utilizes Microsoft’s COM technology to exchange data objects between a client’s custom application and REDI EMS. The following diagram listing the programming languages supported by the REDI EMS API should give you a sense of what’s possible moving forward.



The REDI EMS API is capable of the following:

- Order/Trade Management
  - Ticket Entry | Ticket Update | Ticket Cancellation | Portfolio Trading
  - Order Entry | Order Update | Order Cancellation
  - Doneaway Entry
  - Ticket/Order/Trade (Executions) event message subscription
- Positions & Locate Management
  - Position Data Subscription
  - E-Locate Request & Data Subscription
- Market Data Subscription
  - L1 Subscription for all asset classes
  - L1 Subscription for options series

1.4 Getting Started

If you are interested in using the REDI EMS API, please contact your account manager. Please refer to the 'REDI EMS & API Installation Guide' document once you have been entitled to the REDI EMS API.

Chapter 2 API Classes

2.1 ORDER

2.1.1 Members of ORDER class

Member Name	Data Type	Max Size	Note
Account	String	10	Newer version of REDI EMS supports up to 30 characters.
ClientData	String	36	This field is free text field which persist throughout lifecycle of an order. It is NOT recommended using this field if your firm use portfolio trader or FIX ticket.
CustomerIndicator	String	64	This is free text field that can be set on order; useful for trader instructions allocation information, etc.
Discretion	Float		Discretionary price
DisplayQuantity	Integer		Minimum display order quantity
Exchange	String		Broker (or Exchange) Destination
LocateBroker	String	8	Broker name (or acronym) for short sale orders
MaxDisp	Integer		Maximum display order quantity
Memo	String	30	
PegPrice	Float		Peg Price
Price	Float		Limit Price
PriceType	String		Order type of an order. • Market • Limit • Stop • Stop Limit • Peg Ask • Peg Bid • Peg Mid • Market Close • Limit Close List of valid price types varies. Destination and order entry time are the main drivers.
Quantity	Integer		Order quantity

Member Name	Data Type	Max Size	Note
Side	String		Side of an order • Buy • Buy Cover • Sell • Sell Short • Sell Auto
Symbol	String		REDI EMS Support ticker, Bloomberg, ISIN, SEDOL and RID for Symbol.
Ticket	String		
TicketRef	String		
TIF	String		Time In Force • Day • IOC • FOK • OPG • EXT List of valid TIFs varies. Destination and order entry time are the main drivers.
Warning	Boolean		The purpose of the object member is to suppress REDI EMS pop-ups. It does not suppress (or bypass) all of REDI EMS pop-ups. REDI EMS pop-ups are suppressed when the member is set to False and the user runs into following scenarios: • Invalid order (incorrect side, qty and etc) – orders get rejected • Aggressive price – orders do not get rejected, but go out to the market
SystematicInternalizer	Boolean		(MiFID II field) Systematic Internalizers are investment firms which, on an organized, frequent, systematic, and substantial basis, deal on own account by executing client orders outside a regulated market, MTF, or OTDF without operating a multilateral system
ReducesRisk	Boolean		(MiFID II field) Indicates if a transaction reduces risk
PendingAllocations	Boolean		(MiFID II field) Indication that the trade needs to be allocated (i.e. the trade is not pre-trade allocated)
AggregatedOrders	Boolean		(MiFID II field) Indication the orders have been grouped together

Member Name	Data Type	Max Size	Note
TradingInstruction	String		(MiFID II field) Order-level preference to determine trading venue • None • SI Only • TV Only • TV or SI
AlgoID	String		(MiFID II field) Identifier for an algorithm

2.1.2 Members of ORDER Class

Member Name	Parameter(s)	Description
DisplayMontage	Int x , int y	This method opens up a REDI EMS 'Montage Monitor' window and populates order information in the order object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window Example Following method call opens up a REDI EMS window on the most upper left of user's main monitor. `ordObj.DisplayMontage 1, 1`
DisplayOrderDlg	Int x , int y	This method opens up a REDI EMS 'Classic Order Entry' window and populates order information in the order object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window Example Following method call opens up a REDI EMS window on the most upper left of user's main monitor. `ordObj.DisplayOrderDlg 1, 1`
GetAccountCount	int pnCount	This method gets the count of all available accounts based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output on number of accounts Example Following method call gets count of accounts then saves it in a variable, cnt `ordObj.GetAccountCount cnt`
GetAccountX	int nAccntNum, string Account	This method gets the account name from the account list. Parameter(s) int nAcctNum: input - nth account from the object string Account: output for account name Example Following method call gets 3rd account element from the available order object then saves the account name in a variable, acct. `ordObj.GetAccountX 2, acct`

Member Name	Parameter(s)	Description
GetExchangeCount	int pnCount	This method gets the count of all available broker/exchange destinations based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output on number of destinations Example Following method call gets count of destinations then saves it in a variable, cnt `ordObj.GetExchangeCount cnt`
GetExchangeX	int nExchNum, string Exchange	This method gets the broker/exchange destination name from the exchange list. Parameter(s) int nExchNum: input - nth destination name from the object string Exchange: output for destination name Example Following method call gets 3rd destination element from the available order object then saves the destination name in a variable, exch. `ordObj.GetExchangeX 2, exch`
GetMBFieldCount	int pnCount	This method gets the count of all available custom fields for a broker algo route. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output on number of algo Parameter(s) Example Following method call gets count of algo Parameter(s) then saves it in a variable, *cnt* `ordObj.GetMBFieldCount cnt`
GetMBFieldValueCount	Name, int pnCount	This method returns the count of available MB parameter values. `ordObj.GetMBFieldValueCount Name, cnt`
GetMBFieldValueX	Name, nValueNum, entryName, entryValue	This method returns the values available for a given MB parameter. `ordObj.GetMBFieldValueX Name, nValueNum, entryName, entryValue`

Member Name	Parameter(s)	Description
GetMBFieldX	int nFieldNum, string Name, string Value, int Type	This method gets the algo field details from the order object Parameter(s) int nFieldNum: input - nth algo field from the object string Name: output for a field name of an algo field string Value: output for default value of an algo field int Type: output Example Following method call gets 5th algo field details from the available order object then saves the field name in a variable, Name, default value of the field in a variable, *Value*. Type is a placeholder. `ordObj.GetMBFieldX 4, Name, Value, Type`
GetPriceTypeCount	int pnCount	This method gets the count of all available price types based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output on number of price types Example Following method call gets count of price types then saves it in a variable, *cnt* `ordObj.GetPriceTypeCount cnt`
GetPriceTypeX	int nPxTyNum, string PxType	This method gets the price type name from the order object. Parameter(s) int nPxTyNum: input - nth price type element from the object string PxType: output for price type name Example Following method call gets 3rd price type element from the available order object then saves the price type name in a variable, *PxType*. `ordObj.GetPxTypeX 2, PxType`
GetTIFCount	int pnCount	This method gets the count of all available TIFs(time in force) based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output on number of price types Example Following method call gets count of TIFs then saves it in a variable, *cnt* `ordObj.GetTIFCount cnt`

Member Name	Parameter(s)	Description
GetTIFX	int nTIFNum, string TIF	This method gets the TIF(time in force) name from the order object. Parameter(s) int nTIFNum: input - nth TIF element from the object string TIF: output for price type name Example Following method call gets 3rd price type element from the available order object then saves the price type name in a variable, *TIF*. `ordObj.GetTIFX 2, TIF`
GetTransError	int transID, string err	User can get unique ID of an order, transID, when using `Submit2` method for an order submmission. `GetTransError` method is useful to find out the high level error reason of an order. This method is not for retrieving the order status. Parameter(s) int transID: input – unique ID from order submission via Submit2 method. string err: output for high level error reason for order submission. Example Following method call gets order error reason of an order via `Submit2` method call and saves the reason to a variable; *err*, if order submission was not successful. The variable, err, contains a value, *Unknown*, if the API call for order submission was successful whether the order was accepted by a broker, or rejected by a broker. `ordObj.GetTransError 178, err`
SetOrderKey	string UserID, string OrdRefKey	This method is used for updating an existing order. Parameter(s) string UserID: input – UserID. Not required, you may leave it blank. string OrdRefKey: input – Unique order reference key Example Following method call sets the order reference key of an existing order object to a new order object so that new order object references previous order object for replacing an order. `ordObj.SetOrderKey "", "YSJ20125350"`
SetTFList	string PTListName	This method assigns PT list name to an order object. Parameter(s) string PTListName: input – List name for an PT(portfolio Trade) list. Example `ordObj.SetTFList "PT_LIST_01"`

Member Name	Parameter(s)	Description
SetTFUser	string PTListUser	This method assigns user ID to an order object for a PT List Parameter(s) string PTListUser: input – username for PT List Example `ordObj.SetTFUser "Username"`
Submit	string err	This method is for submitting an order. Parameter(s) string err: output – order submission failure reason if the submission failed Return Boolean: result of the method call whether it is successful or not Example Following method call sends an order and saves a result of the submission in a variable, *rtnVal*. It also saves the error reason in a variable; *err*, if the submission failed. `rtnVal = ordObj.Submit(err)`
Submit2	int TransID, string err	This method is for submitting an order asynchronously. This method gives 2x better performance than 'Submit' method. Parameter(s) string TransID: output – unique positive integer value if order submission was successful. *TransID* is 0 if the order submission failed. string err: output – order submission failure reason if the submission failed Return Boolean: result of the method call whether it is successful or not Example Following method call sends an order and saves a result of the submission in a variable, *rtnVal*. It also saves the unique ID or the order in a variable, TransID, and it saves the error reason in a variable, *err*, if the submission failed. `rtnVal = ordObj.Submit(TransID, err)`

2.2 OPTIONORDER

2.2.1 Members of OPTIONORDER Class

Member Name	Data Type	Max Size	Note
Account	String	10	Newer version of REDI EMS supports up to 30 characters.

Member Name	Data Type	Max Size	Note
ClientData	String	36	This field is free text field which persist throughout lifecycle of an order. It is NOT recommended using this field if your firm use portfolio trader or FIX ticket.
CustomerIndicator	String	64	This is free text field that can be set on order; useful for trader instructions allocation information, etc.
Date	String		Options expiration date in REDI EMS date format • Monthly Option: Jan '16 • Weekly Option: Jan 20 '16
Exchange	String		Broker (or Exchange) Destination
Memo	String	30	
Position	String		Options order position • Open • Close
Price	Float		Limit Price of an order For complex options: • Credit: Negative price • Debit: Positive price
PriceType	String		Order type of an order. • Market • Limit • Stop • Stop Limit • Market Close • Limit Close List of valid price types varies. Destination and order entry time are the main drivers.
Quantity	Integer		Options contract size
Side	String		Side of an order • Buy • Sell
Strike	Float		Options strike price
Ticket	String		
TicketTag	String		
Type	String		Options Type • Put • Call

Member Name	Data Type	Max Size	Note
TIF	String		Time In Force • Day • IOC • FOK • OPG List of valid TIFs varies. Destination and order entry time are the main drivers.

2.2.2 Members of OPTIONORDER Class

Member Name	Parameter(s)	Description
DisplayOptionMontage	Int x , int y	This method opens up a REDI EMS 'Option Montage' window and populates order information in the option order object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window. Example Following method call opens up a REDI EMS window on the most upper left of user's main monitor. `ordObj.DisplayOptionMontage 1, 1`
DisplayOptionOrder	Int x , int y	This method opens up a REDI EMS 'Option Montage' window and populates order information in the option order object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window. Example Following method call opens up a REDI EMS window on the most upper left of user's main monitor. `ordObj.DisplayOptionMontage 1, 1`
DisplayOptionSeries	Int x , int y	This method opens up a REDI EMS 'Option Series' window and populates order information in the option order object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window Example Following method call opens up a REDI EMS window on the most upper left of user's main monitor. `ordObj.DisplayOptionSeries 1, 1`
GetAccountAt	int nIndex	This method gets the account name from the account list. Parameter(s) int nIndex: input - nth account name from the object Return String: Value of an account name Example Following method call gets 3rd account element from the available order object then saves the account name in a variable, *accnt*. `accnt = ordObj.GetAccountAt(2)`

Member Name	Parameter(s)	Description
GetAccountCount	int nCount	This method gets the count of account. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int nCount: output - number of accounts Example Following method call gets the count of accounts then saves it in a variable, *cnt* `ordObj.GetAccountCount cnt`
GetExchangeAt	int nIndex	This method gets the broker/exchange destination name from the exchange list. Parameter(s) int nIndex: input - nth destination name from the object Return String: Value of a destination name Example Following method call gets 3rd destination element from the available order object then saves the destination name in a variable, *exch*. `exch = ordObj.GetExchangeAt(2)`
GetExchangeCount	in nCount	This method gets the count of all available broker/exchange destination names. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output - number of algo Parameter(s) Example Following method call gets count of algo Parameter(s) then saves it in a variable, *cnt* `ordObj.GetExchangeCount cnt`
GetExpirationDateAt	int nIndex	This method gets the options expiration date in REDI EMS format from the expiration date list. Parameter(s) int nIndex: input - nth expiration date from the object Return String: Value of an option expiration date Example Following method call gets 3rd expiration date element from the available order object then saves the expiration date in a variable, *expDate*. `expDate = ordObj.GetExpirationDateAt(2)`

Member Name	Parameter(s)	Description
GetExpirationDatesCount	in nCount	This method gets the count of option expiration dates. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output - number of expiration dates Example Following method call gets count of option expiration dates then saves it in a variable, *cnt* `ordObj.GetExpirationDatesCount cnt`
GetMBFieldCount	int nIndex	This method gets the count of all available custom fields for a broker algo route. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output on number of algo Parameter(s) Example Following method call gets count of algo Parameter(s) then saves it in a variable, *cnt* `ordObj.GetMBFieldCount cnt`
GetMBFieldValueCount	Name, int pnCount	This method returns the count of available MB parameter values. `ordObj.GetMBFieldValueCount Name, cnt`
GetMBFieldValueX	Name, nValueNum, entryName, entryValue	This method returns the values available for a given MB parameter. `ordObj.GetMBFieldValueX Name, nValueNum, entryName, entryValue`
GetMBFieldX	int nFieldNum, string Name, string Value, int Type	This method gets the algo field details from the order object Parameter(s) int nFieldNum: input - nth algo field from the object string Name: output for a field name of an algo field string Value: output for default value of an algo field int Type: output Example Following method call gets 5th algo field details from the available order object then saves the field name in a variable, Name, default value of the field in a variable, *Value*. Type is a placeholder. `ordObj.GetMBFieldX 4, Name, Value, Type`

Member Name	Parameter(s)	Description
GetPriceTypeAt	int pnCount	This method gets the options price type name from the price type list. Parameter(s) int nIndex: input - nth price type from the object Return String: Value of a price type Example Following method call gets 3rd price type element from the available order object then saves the price type name in a variable, *pxType*. `pxType = ordObj.GetPriceTypeAt(2)`
GetPriceTypeCount	int nFieldNum, string Name, string Value, int Type	This method gets the count of price types. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output - number of price types Example Following method call gets count of price types then saves it in a variable, *cnt* `ordObj.GetPriceTypeCount cnt`
GetStrikeAt	int nIndex	This method gets the options strike price from the strike price list. Parameter(s) int nIndex: input - nth strike price from the object Return String: Value of a strike price Example Following method call gets 3rd strike price element from the available order object then saves the strike price in a variable, *strikePx*. `strikePx = ordObj.GetStrikeAt(2)`
GetStrikesCount	int nCount	This method gets the count of options strike prices. The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output - number of strike prices Example Following method call gets count of strike prices then saves it in a variable, *cnt* `ordObj.GetStrikesCount cnt`

Member Name	Parameter(s)	Description
GetTIFAt	nIndex	This method gets the TIF(time in force) from the TIF list. Parameter(s) int nIndex: input - nth TIF from the object Return String: Value of a TIF Example Following method call gets 3rd TIF element from the available order object then saves the TIF in a variable, TIF. `TIF = ordObj.GetTIFAt(2)`
GetTIFCount		This method gets the count of TIFs (time in force). The return of the count is based on the order object with the pre-set filters on the object. Parameter(s) int pnCount: output - number of TIFs Example Following method call gets count of TIFs then saves it in a variable, *cnt* `ordObj.GetTIFCount cnt`
GetTransError	int transID, string err	User can get unique ID of an order, transID, when using Submit2 method for an order submmission. GetTransError method is useful to find out the high level error reason of an order. This method is not for retrieving the order status. Parameter(s) int transID: input – unique ID from order submission via `Submit2` method. string err: output for high level error reason for order submission. Example Following method call gets order error reason of an order via Submit2 method call and saves the reason to a variable; *err*, if order submission was not successful. The variable, err, contains a value, *Unknown*, if the API call for order submission was successful whether the order was accepted by a broker, or rejected by a broker. `ordObj.GetTransError 178, err`
SetOrderKey	string UserID, string OrdRefKey	This method is used for updating an existing order. Parameter(s) string UserID: input – UserID. Not required, you may leave it blank. string OrdRefKey: input – Unique order reference key Example Following method call sets the order reference key of an existing order object to a new order object so that new order object references previous order object for replacing `ordObj.SetOrderKey “”, “YSJ20125350”`

Member Name	Parameter(s)	Description
Submit	string err	This method is for submitting an order. Parameter(s) string *err*: output – order submission failure reason if the submission failed Return Boolean: result of the method call whether it is successful or not Example Following method call sends an order and saves a result of the submission in a variable, *rtnVal*. It also saves the error reason in a variable; *err*, if the submission failed. `rtnVal = ordObj.Submit(err)`
Submit2	string TransID, string err	This method is for submitting an order asynchronously. Parameter(s) string TransID: output – unique positive integer value if order submission was successful. *TransID* is 0 if the order submission failed. string err: output – order submission failure reason if the submission failed Return Boolean: result of the method call whether it is successful or not Example Following method call sends an order and saves a result of the submission in a variable, rtnVal. It also saves the unique ID or the order in a variable, *TransID*, and it saves the error reason in a variable, *err*, if the submission failed. `rtnVal = ordObj.Submit(TransID, err)`

2.3 COMPLEXORDER

2.3.1 Members of COMPLEXORDER class

Member Name	Data Type	Max Size	Description

Strategy	String		Strategy of complex options order. Here're valid values: • Box • Butterfly • Buy-Write • Calendar • Delta Neutral • Diagonal • Married Put • Ratio Orders • Reverse Conversion • Straddle • Strangle • Vertical
CustomerIndicator	String	64	This is free text field that can be set on order; useful for trader instructions allocation information, etc. Please add this right before Submit() method.

2.3.2 Members of COMPLEXORDER class

Member Name	Parameter(s)	Description
GetLegsCount		This method is for getting number of legs a complex option order object. Member Name: SetRatio Parameters Parameter(s) none Return Integer: number of legs of the order object Example Following method call gets number of legs of the order object. By default it will return 1 if there are no legs added to the order object. `rtnVal = ordObj.GetLegsCount`
GetRatio	int nLeg	This method is for getting ratio of a complex option order object. Parameter(s) integer nLeg: input – nth leg of the order object Return integer: ratio of a leg of the order object Example Following method call gets the ratio of a leg from the order object. `rtnVal = ordObj.GetRatio(0)`
GetStrike	int nLeg	This method is for getting strike price of a leg from an order object. Parameter(s) integer nLeg: input – nth leg of the order object Return float: strike price of the nth leg Example Following method call gets the strike price of a leg from the order object. `rtnVal = ordObj.GetStrike(0)`
IsAtLeastOneEquityLeg		This method is for finding out if an order object contains an equity leg. Parameter(s) none Return boolean: True if an order object contains an equity leg, otherwise false Example Following method call finds out if an order object contains an equity leg. `rtnVal = ordObj.IsAtLeastOneEquityLeg`

Member Name	Parameter(s)	Description
IsEquityLeg	int nLeg	This method is for finding out if the nth leg of an order object is an equity leg. Parameter(s) integer nLeg: input – nth leg of the order object Return boolean: True if an order object contains an equity leg, otherwise false Example `rtnVal = ordObj.GetRatio(0)`
IsMonthEnabled	int nLeg	This method is for finding out if the nth leg of an order object is an equity leg. Parameter(s) integer nLeg: input – nth leg of the order object Return boolean: True if an order object contains an equity leg, otherwise false Example `rtnVal = ordObj.GetRatio(0)`
IsNewOrder		This method is for finding out an order object is for entering a new order Parameter(s) none Return boolean: True if an order object is for new order, otherwise false. Example `rtnVal = ordObj.IsNewOrder`
IsOptTypeEnabled	integer nLeg	This method is for finding out if the nth leg of an order object has option type, i.e. put or call. Parameter(s) integer nLeg: input – nth leg of the order object Return boolean: True if an order object contains an option type, otherwise false. Example `rtnVal = ordObj.IsOptTypeEnabled(1)`

Member Name	Parameter(s)	Description
IsPositionEnabled	integer nLeg	This method is for finding out if the nth leg of an order object has option position, i.e. open or close. Parameter(s) integer nLeg: input – nth leg of the order object Return boolean: True if an order object contains an option position, otherwise false. Example `rtnVal = ordObj.IsPositionEnabled(1)`
IsStrikeEnabled	integer nLeg	This method is for finding out if the nth leg of an order object has a strike price. Parameter(s) integer nLeg: input – nth leg of the order object Return boolean: True if an order object contains a strike price, otherwise false. The method call against 0th leg, which is the header of the order, returns nothing. Example `rtnVal = ordObj.IsStrikeEnabled(1)`
RemoveOptionLeg	integer nLeg	This method removes a leg from an order object. Parameter(s) integer nLeg: input – nth leg of the order object Return boolean: True if an let as removed successfully from an order object, otherwise the method call returns nothing. Example `rtnVal = ordObj.RemoveOptionLeg(1)`
SetAccount	int nLeg, string account	This method assigns an account name to nth leg of an order object. Parameter(s) integer nLeg: input – nth leg of the order object string account: input – account name Return none Example `ordObj.SetAccount 1, "TESTACCOUNT"`

Member Name	Parameter(s)	Description
SetExchange	int nLeg, string destination	This method assigns a broker destination name to nth leg of an order object. Please note to choose 0th leg all the time. Parameter(s) integer nLeg: input – nth leg of the order object string destination: input – broker destination name Return none Example `ordObj.SetExchange 0, "APX2 CBOE"`
SetMonth	int nLeg, string expiration	This method assigns an options expiration date to nth leg of an order object. Parameter(s) integer nLeg: input – nth leg of the order object string expiration: input – option expiration date Return none Example `ordObj.SetMonth 1, "Dec '16"`
SetOptType	int nLeg, string optionType	This method assigns an option type (i.e. Put or Call) to nth leg of an order object. Parameter(s) integer nLeg: input – nth leg of the order object string optionType: input – option type. Valid values are *Put* and *Call*. Return none Example `ordObj.SetOptType 1, "Call"`
SetOrderKey	string userId, string orderRefKey	This method assigns an OrderRefKey to an order object so that the user can replace an order. Parameter(s) string userId: input – REDI EMS ID string orderRefKey: input – REDI EMS orderRefKey Return none Example `ordObj.SetOrderKey "parkmm", "YSB1082918"`

Member Name	Parameter(s)	Description
SetPosition	int nLeg, string position	This method assigns an option position (i.e. Open or Close) to nth leg of an order object. Parameter(s) integer nLeg: input – nth leg of the order object string optionType: input – option position. Valid values are *Open* and *Close*. Return none Example `ordObj.SetPosition 1, "Open"`
SetPrice	int nLeg, float price	This method assigns a price to nth leg of an order object. Please note to choose 0th leg all the time. Parameter(s) integer nLeg: input – nth leg of the order object float price: input – order price of the order object Return none Example `ordObj.SetPrice 0, "0.01"`
SetPriceType	int nLeg, string priceType	This method assigns an order type to nth leg of an order object. Please note to choose 0th leg all the time. Parameter(s) integer nLeg: input – nth leg of the order object string priceType: input – order type of the order object. Valid values are Market and Limit. Return none Example `ordObj.SetPriceType 0, "Market"`
SetQuantity	int nLeg, int quantity	This method assigns order quantity to nth leg of an order object. Please note to choose 0th leg all the time. Parameter(s) integer nLeg: input – nth leg of the order object integer quantity: input – order quantity of the order object. Return none Example `ordObj.SetQuantity 0, 15`
SetRatio	int nLeg, int quantity	This method assigns a ratio for the leg quantities (e.g., a header quantity of 2 and a leg ratio quantity of 5 results in a leg quantity of 10). Example `ordObj.SetRatio 1, 5`

Member Name	Parameter(s)	Description
SetSide	int nLeg, string side	This method assigns a side to nth leg of an order object. Parameter(s) integer nLeg: input – nth leg of the order object string side: input – side of the order object. Valid values are *Sell* and *Buy*. Return none Example `ordObj.SetSide 2, "Sell"`
SetStrike	int nLeg, float strikePx	This method assigns a strike price to nth leg of an order object. Parameter(s) integer nLeg: input – nth leg of the order object float strikePx: input – strike price of the order object. Return none Example `ordObj.SetStrike 2, "205"`
SetSymbol	int nLeg, string symbol	This method assigns a underlying product to nth leg of an order object. Please note to choose 0th leg all the time. Parameter(s) integer nLeg: input – nth leg of the order object string symbol: input – underlying product of the order object. Return none Example `ordObj.SetSymbol 0, "IBM"`
SetTIF	int nLeg, string TIF	This method assigns a TIF (Time In Force) to nth leg of an order object. Please note to choose 0th leg all the time. Parameter(s) integer nLeg: input – nth leg of the order object string TIF: input – time in force of the order object. Valid values are Day, IOC, FOK and OPG. Please most brokers support day only. Return none Example `ordObj.SetSymbol 0, "IBM"`

Member Name	Parameter(s)	Description
Submit	string errReason	This method sends order object downstream. Parameter(s) string errReason: output – failure reason for the order submission. Return Boolean result: True if order submission was successfully, otherwise it will return False. Example `rtnVal = ordObj.Submit(myerr)`

2.4 DONEAWAY

2.4.1 Members of Doneaway class

Member Name	Data Type	Max Size	Description
Account	String	10	Newer version of REDI EMS supports up to 30 characters.
Comm	Float		Commission amount
CommType			Commission type. Valid values are: • Per Share • Per Order • Basis Point
Contra	String	4	Contra broker mnemonic
DoneAwayType	String		Doneaway entry type. Valid values are: • US Equity/Futures • Global Equity/Futures
Exchange	String		
Memo	String	30	Brief note for the doneaway entry
OrderQty	Integer		
OrderTime	Time		HH:MM (e.g. 16:20)
Price	Float		Doneaway trade price
Quantity	Integer		Doneaway trade quantity
SD	Date		Settlement Date, MM/DD/YY
Side	String		Valid values are: • BUY • BUY COVER • SELL • SELL SHORT
Solicited	Boolean		An indicator to show whether a doneway trade was solicit or not
Symbol	String		
Ticket	String		

Type	String		Doneaway type. Valid values are: • Corr Clear 7-2 • Corr Clear 7-A • Floor • NASDAQ • OTC Insert • Other • Quick Insert • Upstairs

2.4.2 Members of Doneaway Class

Member Name	Parameter(s)	Description
DisplayDoneawayDlg	int x, int y	This method opens up a REDI EMS 'Doneaway Entry' window and populates order information in the doneaway object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window Example `dawayObj.DisplayDoneawayDlg 1, 1`
Submit	string err	This method is for submitting a doneaway trade. Parameter(s) string err: output – order submission failure reason if the submission failed Return Boolean: result of the method call whether it is successful or not Example Following method call enters a doneaway trade and saves a result of the submission in a variable, *rtnVal*. It also saves the error reason in a variable; *err*, if the submission failed. `rtnVal = dawayObj.Submit(err)`

2.5 Option Doneaway

Member Name	Data Type	Max Size	Description
Account	String	10	Newer version of REDI EMS supports up to 30 characters.
Agency			
Broker	String		Broker code
Capacity	String		• Agency • Principal
CMTA	String		
Comm	Float		Commission amount
CommType	String		Commission type. Valid values are: • Per Contract • Per Order Basis Point
Contra	String	4	Contra broker mnemonic
DisplauOptionDoneAwayDlg			This method opens up a REDI EMS 'Doneaway Entry' window and populates order information in the doneaway object. Parameter(s) int x: Input for horizontal position of new REDI EMS window int y: Input for vertical position of new REDI EMS window Example `dawayObj.DisplayDoneawayDlg 1, 1`
Exchange	String		
ExecPrice	Float		Execution price
ExecQuantity	Integer		Execution quantity
Memo	String	30	Brief note for the doneaway entry
OptionDoneAwayDlgToken			
OptionExpirationMonth	String		Option expiration month
OptionStrikePrice	Float		
OptionType	String		• Call • Put
OutAcronym			

Member Name	Data Type	Max Size	Description
OutClearHouse			
Password			
Position	String		Used to indicate whether the trade is opening or closing a transaction (e.g. buy to close) • Open • Close
RootSymbol	String		Option root (not always the same as the underlier) (e.g. SPXW)
Side	String		• Buy • Sell
Solicited	Boolean		
Submit	String err		This method is for submitting a doneaway trade. Parameter(s) string err: output – order submission failure reason if the submission failed Return Boolean: result of the method call whether it is successful or not Example Following method call enters a doneaway trade and saves a result of the submission in a variable, *rtnVal*. It also saves the error reason in a variable, err, if the submission failed. `rtnVal = dawayObj.Submit(err)`
Symbol	String		Underlier (e.g. use AAPL to enter AAPL equity option doneaways)
Ticket	String		
Time			
timeout			
TradeType	String		Doneaway type: • CMTA • CMTA Out • Floor • Quick Insert
UserID			

2.6 CacheControl

2.6.1 Members of CacheControl Class

Member Name	Data Type	Max Size	Description
CloseQueryOnError	Boolean		This is an attribute to revoke CacheControl instance when there's an error with CacheControl object. It is set to 'False' by default.
OpenQueryAsync	Boolean		CacheControl instance subscribe messages asynchronously when this attribute is set to 'True'. There's no default value.
Password	String		
RowCount	Integer		
UserID	String		

2.6.2 Members of CacheControl Class

Member Name	Parameter(s)	Description

AddWatch	int Type, string Symbol, string Account, string err	This method adds an account or a symbol to a watch list.
		Parameter(s)
		int Type: input – data subscription type. Valid values are:
		0 = L1 Market Data
		1 = L2 Market Data
		2 = P&L
		3 = Firm
		4 = L1 Options Market Data
		5 = Time & Sale
		6 = Historical Time & Sale
		7 = Opra Code
		8 = L1 FX Market Data string Arg1: input – Symbol string Arg2: input - Account
		string err: output – request failure reason
		Return
		Boolean: result of the method call whether it is successful or not
		Example
		L1 market data subscription:
		`rtnVal = obj.AddWatch(0,"IBM","",err)`
		PnL data subscription:
		`rtnVal = obj.AddWatch(2,"IBM","ACCT",err)`

Member Name	Parameter(s)	Description
Cancel	int RowNum, string err	This method cancels an open order. Parameter(s) int RowNum: input – row number of cache control result set string err: output – request failure reason code Return Boolean: result of the method call whether it is successful or not. Example `rtnVal = obj.Cancel(43, err)`
CancelByKey	string UserId, string OrdRefKey, string err	This method cancels an open order. Parameter(s) string UserId: input – user ID string OrdRefKey: input – order ref key string err: output – request failure reason code Return Boolean: result of the method call whether it is successful or not. Example `rtnVal = obj.CancelByKey("Username", "YSJ1042126", err)`

DeleteWatch	int Type, string Symbol, string Account, string err	This method removes an account or a symbol from a watch list.
		Parameter(s)
		int Type: input – data subscription type. Valid values are:
		0 = L1 Market Data
		1 = L2 Market Data
		2 = P&L
		3 = Firm
		4 = L1 Options Market Data
		5 = Time & Sale
		6 = Historical Time & Sale
		7 = Opra Code
		8 = L1 FX Market Data string Arg1: input – Symbol string Arg2: input - Account
		string err: output – request failure reason
		Return
		Boolean: result of the method call whether it is successful or not
		Example
		Remove L1 market data subscription:
		`rtnVal = obj.DeleteWatch(0,"IBM","",err)`
		Remove PnL data subscription:
		`rtnVal = obj.DeleteWatch(2,"IBM","ACCT",err)`

Member Name	Parameter(s)	Description
GetL1Value	string Symbol, string MDFieldName, string TgtVarName	This method gets a value of a field from a data patch Parameter(s) integer RowNum: input – Row number of data subscription String SrcFieldName: input – REDI EMS field name. Refer to 'Appendix A' for the list of available fields. String TgtVarName: output – A variable to hold value of REDI EMS filed string err: output – request failure reason code Return Boolean: result of the method call whether it is successful or not. Example Following example will get the value of Symbol from row # 9 of data patch and stores the value in a variable, 'sym'. `rtnVal = obj.GetCell(9,"Symbol", sym, err)`
GetOptionL1Value	string OPRA, string MDFieldName, string TgtVarName	This method gets L1 data for option Parameter(s) String Symbol: input – OPRA String MDFieldName: input – REDI EMS L1 field name, eg 'CALL_AskPrice'. Refer to 'Appendix A' for the list of available fields. String TgtVarName: output – A variable to hold value of market data field. Return Boolean: result of the method call whether it is successful or not. Example Following example will get the ask price of 'Call option IBM which expires on Feb 5th 2016 with strike px of $125' and stores the value in a variable, CallAP. `rtnVal = obj. GetOptionL1Value("IBM 160205C00125000", "CALL_AskPrice", CallAP)`
Submit	String table, string where, string err	This method starts the real time REDI EMS data subscription. For more details please see 'Appendix A'. Parameter(s) String table: input – Name of a table. Valid values are L1, Position, Message and Elocate. string where: input – where clause string err: output – request failure reason Return Boolean: result of the method call whether it is successful or not Example `rtnVal = obj.Submit(table, where, verr)`

Member Name	Parameter(s)	Description
CacheEvent	int Action, int Row	This method is an event handler where it listens to CacheControl object, and it is used for dispatching real time data including same day snapshot. Parameter(s) Int Action: input – Enum of event type 1: previous data (snapshot) 4: new updates (insert) 5: change to existing data (update) 8: deletion of existing data (delete) 9: release of existing data (release) Int Row: input – number of rows for an event. It is usually 1 for real time data, and it is more than 0 if it is a snapshot when there's activity. Return Boolean: result of the method call whether it is successful or not Example `Sub Query_CacheEvent(ByVal Action As Long, ByVal ROW As Long)` . . . End Sub

2.7 Application

2.7.1 Members of Application Class

Member Name	Data Type	Max Size	Description
FullName	String		It will return full path of **Redi.exe**
IsApplicationReady	Integer		It will return '1' if REDI EMS is ready; 0 otherwise.
TicketEnabled	Integer		It will return '1' if ticket is enabled; 0 otherwise.
TicketEnforced	Integer		It will return '1' if ticket is enforced; 0 otherwise.
UserID	String		It will return REDI EMS ID of current user.

2.7.2 Members of Application Class

Member Name	Parameter(s)	Description
CancelAllOrders	String Side, int Scope, string err	This method cancels all open orders filtered by side and supervisory scope Parameter(s) String Side: input – side of order(s). eg. Sell Integer Scope: input – Scope of supervision where 0 – All orders under current user's supervision 1 – All orders of current user String err: output – error reason Return Boolean: result of the method call whether it is successful or not. Example Following command will cancel all 'BUY' orders which belongs to current user. `rtnVal = obj. CancelAllOrders("Buy", 1, err)`
CancelOrder	String ClientData, String err	This method cancels all open orders filtered by 'ClientData' Parameter(s) String ClientData: input – Client Data String err: output – error reason Return Boolean: result of the method call whether it is successful or not. Example Following command will cancel all orders with 'ClientData' = 'TEST' `rtnVal = obj. CancelOrder("TEST", err`

Member Name	Parameter(s)	Description
CancelOrdersByAcctSymbol	String Account, String Symbol, String err	This method cancels all open orders filtered by account and symbol. Parameter(s) String Account: input – Account Name String Symbol: input - Symbol String err: output – error reason Return Boolean: result of the method call whether it is successful or not. Example Following command will cancel all IBM orders under account name 'ACCT'. `rtnVal = obj.CancelOrdersByAcctSymbol( "ACCT", "IBM", err)`
CancelTicket	String ClientData, String err	This method cancels all open tickets filtered by 'ClientData' Parameter(s) String ClientData: input – Client Data String err: output – error reason Return Boolean: result of the method call whether it is successful or not. Example Following command will cancel all tickets with 'ClientData' = 'TEST' `rtnVal = obj.CancelTicket("TEST", err)`
IsSymbolExist	String Symbol	This method tests if a symbol is available. Parameter(s) String Symbol: input – Symbol Return Integer: Method call returns 1 of a symbol is available; otherwise 0. Example Following command will test if E-mini S&P is available for the current user. `rtnVal = obj.IsSymbolExist("ESH6")`
Quit	String tmpStr	This method terminates RED EMS front-end. Parameter(s) String tmpStr: input – dummy string Return N/A Example `obj.Quit "dummy string"`

Chapter 3 Sample Programs

NOTE: Most of the following sample code will be in VBA. Sample code in other programming languages (e.g. C# or Python) will be provided upon request.

3.1 Order/Trade Management

3.1.1 Equity Order Entry

- VBA Example

```
‘ Create a new order object
Dim hOrder As New Order

‘ Set order attributes to the order object
hOrder.symbol = “ZVZZT”
hOrder.Side = “Buy”
hOrder.Quantity = “100”
hOrder.Exchange = “DEMO DMA”
hOrder.PriceType = “Limit”
hOrder.Price = 0.01
hOrder.TIF = “Day”
hOrder.Account = “TESTACCT”
hOrder.Ticket = “Bypass”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

- Python Example

```
# Load COM interface
import win32com.client

# Equity Order Entry Example
o = win32com.client.Dispatch(“REDI.ORDER”)
o.Side = “Buy”
o.symbol = “SBUX”
o.Exchange = “DEMO DMA”
o.Quantity = “1”
o.PriceType = “Market”
o.TIF = “Day”
o.Account = “TESTACCT”
o.Ticket = “Bypass”

# Prepare a variable which can handle returned values from submit method of the order
object.
msg = win32com.client.VARIANT(win32com.client.pythoncom.VT_BYREF | win32com.client.
pythoncom.VT_VARIANT, None)

# Send an options order
```

```
result = o.Submit(msg)

print result # 'True' if order submission was successful
print msg.value # message from sumbit
```

- C# Example

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text;
using System.Threading.Tasks; using Lib;

namespace REDIEquityOrderEntry
{
class Program
{
static void Main(string[] args)
{
ORDER hOrder = new ORDER(); Object err = null;
Object result;

hOrder.Side = "Buy"; hOrder.Symbol = "ZVZZT"; hOrder.Quantity = "1"; hOrder.PriceType = "Market";
        hOrder.Exchange = "DEMO DMA"; hOrder.TIF = "Day"; hOrder.Ticket = "Bypass"; hOrder.Account =
        "TESTACCT";

result = hOrder.Submit(err);

System.Console.WriteLine(result + " " + err); Console.ReadLine();
}
}
}
```

3.1.2 Option Order Entry

```
' Create a new option order object
Dim hOrder As New OPTIONORDER

' Set order attributes to the order object
hOrder.symbol = "SBUX"
hOrder.Type = "Call"
hOrder.Date = "Apr '16"
hOrder.Side = "Buy"
hOrder.Strike = "40"
hOrder.Quantity = "1"
hOrder.Position = "Open"
hOrder.Exchange = "DEM1 DMA"
hOrder.PriceType = "Limit"
hOrder.Price = 0.01
hOrder.TIF = "Day"
hOrder.Account = "TESTACCT"
hOrder.Ticket = "Bypass"

Dim errStr As Variant
Dim rtnVal As Variant
```

```
‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.3 Complex Option Order Entry

```
Dim hOrder As New COMPLEXORDER

‘ Complex options order header
hOrder.Strategy = “Buy-Write”
hOrder.SetSymbol 0, “IBM”
hOrder.SetExchange 0, “DEM2 DMA”
hOrder.SetPriceType 0, “Market”
hOrder.SetTIF 0, “Day”
hOrder.SetQuantity 0, 1

‘ Leg 1 of Buy-Write leg (First leg has to be an option order)
hOrder.SetSide 1, “Buy”
hOrder.SetPosition 1, “Close”
hOrder.SetOptType 1, “Call”
hOrder.SetMonth 1, “Feb ‘16”
hOrder.SetStrike 1, “120.00”
hOrder.SetAccount 1, “TESTACCT”

‘ Leg 2 of Buy-Write leg (Equity Leg)
hOrder.SetSide 2, “Sell”
hOrder.SetAccount 2, “ TESTACCT “

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.4 Ticket

```
‘ Create a new order object
Dim hOrder As New Order

‘ Set order attributes to the order object
hOrder.symbol = “ZVZZT”
hOrder.Side = “Buy”
hOrder.Quantity = “100”
hOrder.Exchange = “*ticket”
hOrder.PriceType = “Market”
hOrder.TIF = “Day”
hOrder.Account = “TEST”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.5 Portfolio Trading List Entry

The following example illustrates how to construct portfolio list.

NOTE: This example is not for waving out an basket out to brokers.

```
‘ Create a new order object
Dim hOrder As New Order

‘ Set order attributes to the order object
hOrder.symbol = “ESZ6”
hOrder.Side = “Buy”
hOrder.Quantity = 10
hOrder.Exchange = “*ticket”
hOrder.Account = “TESTACCT”
hOrder.SetTFList “PT_FUT_01”
hOrder.SetTFUser “parkmm”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.6 Algorithm Order Entry

```
‘ Create a new order object
Dim hOrder As New Order

‘ Set order attributes to the order object
hOrder.symbol = “ZVZZT”
hOrder.Side = “Buy”
hOrder.Quantity = “100”
hOrder.Exchange = “DEMO ALGO”

‘ Set algo strategy
hOrder.PriceType = “VWAP DEMO”
hOrder.Price = 0.01

hOrder.TIF = “Day”
hOrder.Account = “00999900”
hOrder.Ticket = “Bypass”

‘ Set custom algo fields
hOrder.SetNewVariable “(MB) Start Time”, “155500”
hOrder.SetNewVariable “(MB) End Time”, “160000”
hOrder.SetNewVariable “(MB) Exec Style”, “Aggressive”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
Debug.Print errStr
```

3.1.7 Order Entry – Linking to a Ticket

```
‘ Create a new order object
Dim hOrder As New Order

‘ Set order attributes to the order object
hOrder.symbol = “ZVZZT”
hOrder.Side = “Buy”
hOrder.Quantity = “100”
hOrder.Exchange = “DEMO DMA”
hOrder.PriceType = “Limit”
hOrder.Price = 0.01
hOrder.TIF = “Day”
hOrder.Account = “TESTACCT”

‘ where NOF is branch code
‘ and 1032 is branch sequence number of a ticket
hOrder.Ticket = “NOF 1032”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.8 Doneaway Entry

```
‘ Create a new doneaway object
Dim hOrder As New Doneaway

‘ Set doneaway attributes to the order object
hOrder.DoneAwayType = “US Equity/Futures”
hOrder.Exchange = “OTC”
hOrder.Type = “Quick Insert”
hOrder.Side = “SELL”
hOrder.Quantity = 12300
hOrder.Symbol = “ZVZZT”
hOrder.Price = 100.01
hOrder.Account = “00999900”
hOrder.Ticket = “Bypass”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send doneaway object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.9 Order Update

```
‘ Create a new order object
Dim hOrder As New Order

‘ Set order attributes to the order object
hOrder.SetOrderKey “parkmm”, “YSJ1025332”
hOrder.Quantity = “200”

Dim errStr As Variant
Dim rtnVal As Variant

‘ Send order object to REDI EMS
rtnVal = hOrder.Submit(errStr)
```

3.1.10 Order Cancellation

3.1.10.1 Cancelling an Order By ‘OrderRefKey

```
Public WithEvents OrderQuery As CacheControl

Sub CancelOrder()
Set OrderQuery = New CacheControl Dim verr, table, where

table = “Message”
where = “Status==0 || Status==1 || Status==2 || Status==3”
OrderQuery.Submit table, where, verr

‘ Cancel Order
OrderQuery.CancelByKey “parkmm”, “YSJ1069332”, verr End Sub
```

3.1.10.2 Cancelling Orders By ‘ClientData’

```
Dim App As New RediLib.Application
Dim verr, rtnVal

rtnVal = App.CancelOrder(“myClientDataStr”, err)
```

3.1.10.3 Cancelling Tickets By ‘ClientData’

```
Dim App As New RediLib.Application
Dim verr, rtnVal

rtnVal = App.CancelTicket(“myClientDataStr”, err)
```

3.1.10.4 Cancelling Orders By 'Side'

```
Dim App As New RediLib.Application
Dim verr, rtnVal

rtnVal = App.CancelAllOrders("Sell", verr)
```

3.1.10.5 Cancelling Orders By 'Account' and 'Symbol'

```
Dim App As New RediLib.Application
Dim verr, rtnVal

rtnVal = App.CancelOrdersByAcctSymbol("TESTACCT", "IBM", err)
```

3.1.11 Reference Data Subscription[1]

3.1.11.1 Getting Exchange (a.k.a) List for Equity

```
Dim hOrder As New Order Dim RetVal, I, cnt, exch

hOrder.symbol = "IBM" ' This will limit destination list RetVal = hOrder.GetExchangeCount(cnt)

For I = 0 To cnt - 1 hOrder.GetExchangeX I, exch Debug.Print exch
Next I
```

3.1.11.2 Getting PriceTypes for Equity

```
Dim hOrder As New Order Dim RetVal, I, cnt, prType

hOrder.Exchange = "BETA DMA"
RetVal = hOrder.GetPriceTypeCount(cnt)

For I = 0 To cnt - 1 hOrder.GetPriceTypeX I, prType Debug.Print prType
Next I
```

3.1.11.3 Getting PriceTypes for Equity

```
Dim hOrder As New Order Dim RetVal, I, cnt, TIF

hOrder.Exchange = "BETA DMA" hOrder.PriceType = "Limit" RetVal = hOrder.GetTIFCount(cnt)

For I = 0 To cnt - 1 hOrder.GetTIFX I, TIF Debug.Print TIF
Next I
```

[1] Use the OPTIONORDER object to get the list of destinations for options flow.

3.1.11.4 Getting Accounts for Equity

```
Dim hOrder As New Order Dim RetVal, I, cnt, acct

hOrder.symbol = “IBM”
RetVal = hOrder.GetAccountCount(cnt)

For I = 0 To cnt - 1 hOrder.GetAccountX I, acct Debug.Print acct
Next I
```

3.1.11.5 Getting Broker Algorithm Parameters for Equity

```
Dim I Dim cnt
Dim errCode Dim output1 Dim output2 Dim output3
Dim hOrder As New Order

hOrder.symbol = “ZVZZT” hOrder.Side = “Buy” hOrder.Quantity = 1 hOrder.Exchange = “DEMO ALGO”
       hOrder.PriceType = “VWAP DEMO”

‘Get algo fields of VWAP hOrder.GetMBFieldCount cnt For I = 0 To cnt - 1
hOrder.GetMBFieldX I, output1, output2, output3 Debug.Print “”
Debug.Print “fNum : “ & I Debug.Print “fName : “ & output1
Debug.Print “fValue: “ & output2 ‘Default value if exists Debug.Print “fType : “ & output3
Next I
```

3.1.11.6 Get Options Expiration Dates

```
Dim hOrder As New OPTIONORDER hOrder.symbol = “SBUX” hOrder.Type = “Call”

Dim cnt, I hOrder.GetExpirationDatesCount cnt

For I = 0 To cnt – 1
Debug.Print hOrder.GetExpirationDateAt(I) Next I
```

3.1.11.7 Get Options Strike Prices

```
Dim hOrder As New OPTIONORDER hOrder.symbol = “SBUX” hOrder.Type = “Call” hOrder.Date = “Apr ‘16”

Dim cnt, I hOrder.GetStrikesCount cnt

For I = 0 To cnt – 1
Debug.Print hOrder.GetStrikeAt(I) Next I
```

3.1.12 Fill Message Subscription

```
Option Explicit
Dim R4S_Sht As String

‘ Define an event object that will subscribe data from REDI EMS Public WithEvents DataSubscription As
       CacheControl
Dim MessageQuery

‘---------------------------------------------------------------------
‘ This VBA subroutine is the main function to invoke data subscription ‘-----------------------------------
       ---
-----------------------------
Sub OpenMessageTable()

Dim verr As Variant Dim vtable As Variant Dim vwhere As Variant Dim MSG As Variant

If (MessageQuery) Then ‘ A query was previously set DataSubscription.Revoke verr ‘ clear it before setting
       new
verr = Empty
Set MessageQuery = Nothing End If

‘Create new data subscription object if it doesn’t exist If DataSubscription Is Nothing Then
Set DataSubscription = New CacheControl
End If

‘ Message table contains audit trail of all orders
‘ that are associates to current user, and users that ‘ current user supervises
vtable = “Message”

‘ Status=1 : orders that are in partially filled state ‘ Status=2 : orders that are in fully filled state
       vwhere
= “Status==1 || Status==2”

MessageQuery = DataSubscription.Submit(vtable, vwhere, verr)

End Sub

‘---------------------------------------------------------------------
‘ This VBA subroutine is the event handler function which listens to ‘ the event object that is invoked by
       the
other subroutine,
‘ OpenMessageTable()
‘---------------------------------------------------------------------
Sub DataSubscription_CacheEvent(ByVal Action As Long, ByVal ROW As Long)

Dim I As Integer
Dim BrSeq As Variant Dim Side As Variant
Dim ExecQuantity As Variant Dim symbol As Variant
Dim ExecPrice As Variant Dim Status As Variant Dim myerr As Variant
Dim Account As Variant Dim Quantity As Variant Dim RefNum As Variant Dim PriceType As Variant
Dim OrderRefKey As Variant Dim Exchange As Variant Dim Price As Variant
Dim TIF As Variant Dim Memo As Variant
Dim MsgType As Variant Dim OrdDesc As String

‘ Action=1 returns the snapshot of what happened throughout the day If (Action = 1) Then
```

```
For I = 0 To ROW - 1
DataSubscription.GetCell I, “BrSeq”, BrSeq, myerr DataSubscription.GetCell I, “Side”, Side, myerr
DataSubscription.GetCell I, “ExecQuantity”, ExecQuantity, myerr DataSubscription.GetCell I, “ExecPrice”,
ExecPrice, myerr DataSubscription.GetCell I, “symbol”, symbol, myerr DataSubscription.GetCell I,
“Account”, Account, myerr DataSubscription.GetCell I, “Quantity”, Quantity, myerr
DataSubscription.GetCell I, “RefNum”, RefNum, myerr DataSubscription.GetCell I, “Status”, Status, myerr
DataSubscription.GetCell I, “PriceType”, PriceType, myerr DataSubscription.GetCell I, “Price”, Price,
myerr DataSubscription.GetCell I, “Memo”, Memo, myerr
DataSubscription.GetCell I, “TIF”, TIF, myerr DataSubscription.GetCell I, “OrderRefKey”, OrderRefKey, myerr
DataSubscription.GetCell I, “MsgType”, MsgType, myerr

myArr(arNum,1)= ExecQuanaty Next I

‘ Action=4 caches real time updates ElseIf (Action = 4) Then
DataSubscription.GetCell ROW, “BrSeq”, BrSeq, myerr DataSubscription.GetCell ROW, “Side”, Side, myerr
DataSubscription.GetCell ROW, “ExecQuantity”, ExecQuantity, myerr DataSubscription.GetCell ROW,
“ExecPrice”, ExecPrice, myerr DataSubscription.GetCell ROW, “symbol”, symbol, myerr
DataSubscription.GetCell ROW, “Account”, Account, myerr DataSubscription.GetCell ROW, “Quantity”,
Quantity, myerr DataSubscription.GetCell ROW, “RefNum”, RefNum, myerr DataSubscription.GetCell ROW,
“Status”, Status, myerr DataSubscription.GetCell ROW, “PriceType”, PriceType, myerr
DataSubscription.GetCell ROW, “Price”, Price, myerr DataSubscription.GetCell ROW, “Memo”, Memo, myerr
DataSubscription.GetCell ROW, “TIF”, TIF, myerr DataSubscription.GetCell ROW, “OrderRefKey”,
OrderRefKey, myerr DataSubscription.GetCell ROW, “MsgType”, MsgType, myerr
End If
End Sub
```

3.2 Positions & Locate Management

3.2.1 Position Monitoring

```
Option Explicit
Dim R4S_Sht As String

‘ Define an event object that will subscribe data from REDI EMS Public WithEvents PosQuery As CacheControl
Dim PosMsgQuery

‘-------------------------------------------------------------------
‘ This VBA subroutine is the main function to invoke data subscription ‘-----------------------------------
      ---------------------------------
Sub OpenPositionTable()

Dim vreturn As Variant Dim verr As Variant Dim vTable As Variant Dim vwhere As Variant

If PosQuery Is Nothing Then

‘ Create new event handler object Set PosQuery = New CacheControl

‘ Position table contains all position information vTable = “Position”
vwhere = “true”
PosMsgQuery = PosQuery.Submit(vTable, vwhere, verr)

End If

‘ Start watching position for account, TESTACCT vreturn = PosQuery.AddWatch(2, “”, “TESTACCT”, verr)
```

```
End Sub

‘---------------------------------------------------------------------
‘ This VBA subroutine is the event handler function which listens to ‘ the event object that is invoked by
       the other subroutine,
‘ OpenPositionTable()
‘---------------------------------------------------------------------
Sub PosQuery_CacheEvent(ByVal Action As Long, ByVal ROW As Long)

Dim mycolumn As Variant Dim Account As Variant Dim Symbol As Variant
Dim SymbolPosition As Variant Dim AveragePrice As Variant Dim DisplaySymbol As Variant Dim SharesSold As
       Variant
Dim OpraSym As Variant Dim BuyingPower As Variant
Dim excessEquity As Variant Dim AmountBought As Variant Dim AmountSold As Variant Dim OpenPosition As Variant
Dim ProductType As Variant Dim myerr As Variant
Dim I As Integer

‘ Action=1 returns the snapshot of what happened throughout the day If (Action = 1) Then ‘Event is caused by
       submit

For I = 0 To ROW - 1
PosQuery.GetCell I, “Account”, Account, myerr PosQuery.GetCell I, “Symbol”, Symbol, myerr PosQuery.GetCell I,
       “SymbolPosition”, SymbolPosition, myerr PosQuery.GetCell I, “DisplaySymbol”, DisplaySymbol, myerr
       PosQuery.GetCell I, “AmountBought”, AmountBought, myerr PosQuery.GetCell I, “AmountSold”, AmountSold,
       myerr PosQuery.GetCell I, “OpenPosition”, OpenPosition, myerr PosQuery.GetCell I, “ProductType”,
       ProductType, myerr
Next I Else
PosQuery.GetCell ROW, “Account”, Account, myerr PosQuery.GetCell ROW, “Symbol”, Symbol, myerr
       PosQuery.GetCell ROW, “SymbolPosition”, SymbolPosition, myerr PosQuery.GetCell ROW, “DisplaySymbol”,
       DisplaySymbol, myerr PosQuery.GetCell ROW, “AmountBought”, AmountBought, myerr PosQuery.GetCell ROW,
       “AmountSold”, AmountSold, myerr PosQuery.GetCell ROW, “OpenPosition”, OpenPosition, myerr
       PosQuery.GetCell ROW, “ProductType”, ProductType, myerr

End If End Sub
```

3.2.2 E-Locate Monitoring

```
Option Explicit
Dim R4S_Sht As String

‘ Define an event object that will subscribe data from REDI EMS Public WithEvents ELocateQuery As
       CacheControl
Dim MessageQuery

‘---------------------------------------------------------------------
‘ This VBA subroutine is the main function to invoke data subscription ‘-----------------------------------
       --------------------------------
Sub OpenElocateTable() Dim verr As Variant Dim vtable As Variant Dim vwhere As Variant

If (MessageQuery) Then ‘ A query was previously set ELocateQuery.Revoke verr ‘ clear it before setting new
       verr = Empty
Set MessageQuery = Nothing
End If

If ELocateQuery Is Nothing Then
Set ELocateQuery = New CacheControl End If

‘Elocate table contains all Elocate information vtable = “Elocate”
```

```
‘wwhere = “Symbol=='IBM'” ‘Use this if you are interested in IBM only vwhere = “”
MessageQuery = ELocateQuery.Submit(vtable, vwhere, verr)
End Sub

‘---------------------------------------------------------------------
‘ This VBA subroutine is the event handler function which listens to ‘ the event object that is invoked by
      the other subroutine,
‘ OpenElocateTable()
‘---------------------------------------------------------------------
Sub ELocateQuery_CacheEvent(ByVal Action As Long, ByVal ROW As Long)

Dim I As Integer
Dim myerr As Variant Dim SYMBOL As Variant Dim TotalQTY As Variant Dim USEDQTY As Variant
Dim TotalAvailQTY As Variant

‘ Action=1 returns the snapshot of what happened throughout the day If (Action = 1) Then

For I = 0 To ROW – 1

OrderQuery.GetCell I, “SYMBOL”, SYMBOL, myerr OrderQuery.GetCell I, “TotalQTY”, TotalQTY, myerr
      OrderQuery.GetCell I, “USEDQTY”, USEDQTY, myerr OrderQuery.GetCell I, “TotalAvailQTY”, TotalAvailQTY,
      myerr
Next I End If End Sub
```

3.3 Market Data Subscription

3.3.1 L1 Market Data Subscription

3.3.1.1 Getting L1 Market Data Stream

```
Option Explicit
Dim R4S_Sht As String

‘ Declaration for L1 subscription
Private WithEvents L1Cache As RediLib.CacheControl

‘---------------------------------------------------------------------
‘ This VBA subroutine is the main function to invoke L1 subscription
Sub StartL1()

‘ Instantiate event handler If L1Cache Is Nothing Then
Set L1Cache = New RediLib.CacheControl End If

Dim vTable As String Dim vwhere As String Dim vreturn As Variant Dim verr As Variant Dim tmpVal As Variant
      Dim Symbol As String

L1Cache.OpenQueryAsync = True L1Cache.CloseQueryOnError = True

‘ L1 tables contains all level 1 market data that current user is ‘ is entitled to.
vTable = “L1”

vwhere = “true”
tmpVal = L1Cache.Submit(vTable, vwhere, verr)

‘ Start watching all the L1 market data updates for IBM tmpVal = L1Cache.AddWatch(0, “VOD.L”, “”, verr)
```

```
End Sub

‘---------------------------------------------------------------------
‘ This VBA subroutine is the event handler function which listens to ‘ the event object that is invoked by
       the other subroutine,
‘ L1Cache()
‘---------------------------------------------------------------------
Sub L1Cache_CacheEvent(ByVal Action As Long, ByVal ROW As Long)

Dim L1_Symbol As Variant Dim L1_AP As Variant
Dim L1_BP As Variant Dim L1_LP As Variant Dim L1_CP As Variant Dim L1_OP As Variant Dim L1_VWAP As Variant
Dim verr As Variant

If Action = 1 Or Action = 5 Then

L1Cache.GetCell ROW, “Symbol”, L1_Symbol, verr L1Cache.GetCell ROW, “Ask”, L1_AP, verr L1Cache.GetCell ROW,
       “Bid”, L1_BP, verr L1Cache.GetCell ROW, “Last”, L1_LP, verr L1Cache.GetCell ROW, “Close”, L1_CP, verr
       L1Cache.GetCell ROW, “Open”, L1_OP, verr L1Cache.GetCell ROW, “VWAP”, L1_VWAP, verr

‘ stdout to VBA immediate window to see if this works Debug.Print “Time : “ & Now()
Debug.Print “Symbol : “ & L1_Symbol Debug.Print “Ask Px: “ & L1_AP Debug.Print “Bid Px: “ & L1_BP Debug.Print
       “Last Px: “ & L1_LP Debug.Print “Cls Px: “ & L1_CP Debug.Print “Open Px: “ & L1_OP Debug.Print “Vwap
       Px: “ & L1_VWAP Debug.Print “”
End If End Sub
```

3.3.1.2 Getting Futures L1 for a Specific Field

```
Dim L1Cache As New RediLib.CacheControl
Dim vTable, vwhere, verr, tmpVal, askPx As Variant

vTable = “L1”
vwhere = “true”
tmpVal = L1Cache.Submit(vTable, vwhere, verr)
L1Cache.GetL1Value “ESH6”, “Ask”, askPx
‘ stdout ask price
Debug.Print askPx
```

NOTE: This code is valid for equity products.

3.3.1.3 Getting Options L1 for a Specific Field

```
Dim L1Cache As New RediLib.CacheControl
Dim vTable, vwhere, verr, tmpVal, call_askPx As Variant
vTable = “L1”
vwhere = “true”
tmpVal = L1Cache.Submit(vTable, vwhere, verr)

‘ Get ask price for: Call IBM 2016/02/04 StrikePx $125.00
tmpVal = L1Cache. GetOptionL1Value(“IBM 160205C00125000”, “CALL_AskPrice”, call_askPx)
Debug.Print call_askPx
```

NOTE: This code is valid for equity products.

Appendix A Data Fields

The CacheControl object enables users to subscribe to real time data from REDI EMS's back-end via the REDI EMS client. To request this data, users need to supply the table name together with the 'where' statement. The following is a list of available fields for REDI EMS data tables and options for the aforementioned 'where' statement.

A.1 L1 Market Data Subscription

Table Name: L1		
Available Fields		
AnnHi	AnnLo	Ask
AskAtClose	AskDir	AskExchange
AskPrice	AskSize	AuctionPrice
AuctionVol	BASpread	Bid
Bidask	BidAtClose	BidDir
BidExchange	BidPrice	BidSize
BidTick	BrokerName	CALL_AskPrice
CALL_AskSize	CALL_BidPrice	CALL_BidSize
CALL_CDate	CALL_CPrice	CALL_DayHigh
CALL_DayLow	CALL_DayVol	CALL_IndicativeAskPrice
CALL_IndicativeAskSize	CALL_IndicativeBidPrice	CALL_IndicativeBidSize
CALL_LastSale	CALL_LPrice	CALL_OPrice
CALL_QCond	CALL_SettleDate	CALL_SettlePrice
CALL_YAPrice	CALL_YBPrice	Close
ConditionList	ConsolidatedClose	ConsolidatedOpen
Cross	Currency	DayHi
DayLo	DayVolume	DayVWAP
Imbalance	ImbalTime	IsHalted
Last	LastAtClose	LastExch
LastVolume	MktVWAP	Open

Table Name: L1		
PUT_AskPrice	PUT_AskSize	PUT_BidPrice
PUT_BidSize	PUT_CDate	PUT_CPrice
PUT_DayHigh	PUT_DayLow	PUT_DayVol
PUT_IndicativeAskPrice	PUT_IndicativeAskSize	PUT_IndicativeBidPrice
PUT_IndicativeBidSale	Put_LastSale	PUT_LPrice
PUT_OPrice	PUT_QCond	PUT_SettleDate
PUT_SettlePrice	PUT_YAPrice	PUT_YBPrice
TodaysClose	Volume	VWAP
YestHi	YestLo	
Where Options:		
There are many ways to query. Please refer to the sample code provided above.		

A.2 Order Activity Data Subscription

Table Name: Message		
Available Fields		
Account	AccountAlias	ActClearNum
Agency	AllOrNone	BID
BranchCode	BranchSeq	ClientData
CrossSession	Currency	Date
DispSize	EntryUserID	Exchange
ExchangeType	ExecPrice	ExecQuantity
LeavesQty	Memo	MessageSource
MsgType	OrderRefKey	OrdStat
ParentOrdRefKey	PrefMarketMaker	Price
PriceType	Quantity	RefNum

Table Name: Message		
RegionGroup	RIC	Sector
Sedol	Side	StopPrice
Symbol	TIF	Time
UnderlyingSymbol		

Where Options:
MsgType
0: Invalid
1: Outbound Order
2: Outbound Cancel
3: Outbound Cancel Replace
10: Order
13: Reject
14: Execution
15: Administration
16: Pending Order Delete
17: Execution Changed
18: Order Changed
19: Information
Status
0: Open
1: Partial
2: Complete
3: Canceled
4: Reject
5: Deleted

Table Name: Message
6: Failed
7: Pending Reroute
8: Pending

A.3 Position Data Subscription

Table Name: Position		
Available Fields		
Account	AccountName	AccountType
Active	AmountBought	AmountSold
AnnualHigh	AnnualLow	BuyingPower
CashAdjustment	Change	Concentration
HedgeRatio	Imbalance	LedgerBalance
ListingStatus	LoIndPrice	LongValue
MarketCap	MarketCategory	Message
Name	NetChange	NetValue
NewsTime	Open	OpenPosition
OutstandingShares	PandL	PendingAmountBought
PendingAmountSold	PendingPandL	PendingPosition
PendingRequirements	PendingSharesBought	PendingSharesSold
PendingValue	PercentChange	Position
ProductType	Requirements	RunningBalance
SharesBought	SharesSold	ShortValue
Symbol	Time	Value
Where Options:		
There are many ways to query. Please refer to the sample code provided above.		
0: Invalid		

A.4 Locate Data Subscription

Table Name: Position		
Available Fields		
Account	AccountName	AccountType
Active	AmountBought	AmountSold
Where Options:		
There are many ways to query. Please refer to the sample code provided above.		

Document Version: 2.4
Date of issue: 3 December 2019

REFINITIV™