Specification of CASH Algo API

Table of Contents

Purpose

This API would list out the available methods to call in order to perform required actions. Message specification would be provided at a later stage.

General format of strategy file

In order to import required modules into strategy file, the following format shall be followed. This marks that an import of `cashAlgoAPI` is required, while an object of manager should been initialized in order to perform any actions with external parties.

```
import cashAlgoAPI

if __name__ == '__main__':
    global mgr
    mgr = cashAlgoAPI.CASHOrderManager("foo")
    mgr.start()
```

Initialization

CASHOrderManager(name, , username, password, beginDate, endDate)

name : Your strategy name
username : Username provided by CASH
password : Password provided by CASH related to the username above
beginDate : Start date on running this strategy. This would affect the market data subscription period also.
endDate : End date on running this strategy. This would affect the market data subscription period also.
returns : Instance of CASHOrderManager for calling

This is the manager to handle the communication between your strategy and the server in CASH. It has to be initialized at the first place, and to invoke `start` method right after initialization. Code within this object would run in another thread which would not affect your code.

Internal

This method would send a message to server logging credential information. Format required in sequence is shown as follows:-
timestamp : API timestamp
message : Hard-coded as “loginfeed”
username : Username provided by CASH
password : Password provided by CASH related to the username above

Market Data API

subscribeMarketData([callback], market, productCode)

callback : Method to be called in strategy code. The method declaration should be in the following format. Only the last callback function will be triggered.

```
def [callback](marketData):
```

marketData : Latest market data object. Please refer below

returns : None

market : Market ID of the product listed

productCode : The predefined code representing the product

Market data object is a python object with properties available as follows:-

timestamp : Server timestamp

productCode : The predefined code representing the product

lastPrice : Latest traded price

lastVolume : Latest traded volume, 0 means bid ask update only

bidQueue [0..10] : Bid queue, first item is the best bid / ask

askQueue [0..10] : Ask queue, first item is the best bid / ask

This is a subscription method to CASH API where the callback method would be invoked once there are any updates of about the product code. Market data includes both price and volume in each bid and ask queue.
Please note that price / size value with 999999 means not available.

Internal

This method would send a message to server requesting market data in a specified date range. Format required in sequence is shown as follows:-

timestamp : API timestamp

message : Hard-coded as “subscription”

market : The market code

productCode : The predefined code representing the product

beginDate : The start date of market data requested by strategy. The format is “YYYYMMDD”

endDate : The end date of market data requested by strategy. The format is “YYYYMMDD”

Market Access API

insertOrder(order)

order : Order object

returns : None

This is the method to call when the strategy decides to insert an order. It has to be called within the callback method subscribing market data. Otherwise the behavior of the order creation request would be unpredictable. Further order feedback will be returned from order feed.

Order object is a python object with properties available as follows:-

timestamp : API timestamp given by market data
market : Market ID of the product listed
productCode : The predefined code representing the product
orderID : Unique string indicating the order. It should be provided by the strategy
price : Price of the order
volume : Number of contract(s) desired to trade in the order
openClose : Indicate whether this order refers to an open or close trade. Empty if order action is “delete”
buySell : 1 – Buy / 2 – Sell
action : “insert” / “delete”. This indicates whether the request inserts or deletes an order
orderType : “limit_order”. Empty if action is “delete”
orderValidity : “today”. Empty if action is “delete”

getWorkingOrders() (Real trade only)

returns : Array of order feed

This is the method to call when the strategy wants to know how many orders are active in the market, containing its details. It replies the working orders in the format of ("orderfeed") with source set as "1”.

registerTradeFeed([callback])

callback : Method to be called in strategy code. The method declaration should be in the following format.

```
def [callback](tradeFeed):
```

tradefeed : Trade feed object. Please refer below
returns : None

This is the method to call when the strategy would like to subscribe the updates of trades. The tradeFeed parameter in the callback method would contain information of the trade.

TradeFeed object is a python object with properties available as follows:-

timestamp : Server timestamp
market : Market ID of the product listed

productCode : The predefined code representing the product
orderID : Unique string indicating the order. It should be provided by the strategy
price : Traded price
volume : Volume traded
buySell : 1 – Buy / 2 – Sell
source : 0 - "Market" for real-time updated from exchange; 1 - "MessageReply"

registerOrderFeed([callback])

callback : Method to be called in strategy code. The method declaration should be in the following format.

```
def [callback](orderFeed):
```
orderfeed : Order feed object. Please refer below
returns : None

This is the method to call when the strategy would like to subscribe the updates of orders status. The orderFeed parameter in the callback method would contain specific order details.

OrderFeed object is a python object with properties available as follows:-
timestamp : Server timestamp
market : Market ID of the product listed
productCode : The predefined code representing the product
orderID : Unique string indicating the order. It should be provided by the strategy
price : Price of the order
volume : Number of contract(s) desired to trade in the order
openClose : Indicate whether this order refers to an open or close trade. Empty if order action is “delete”
buySell : 1 – Buy / 2 – Sell
volumeFilled : Volume of order filled
deleted : 1 – if order is deleted; 0 - otherwise
status : 0 – OK; 1 - Error
errorDescription : Empty if order status is OK; Error message in string if status is 1
source : 0 - "Market" for real-time updated from exchange; 1 - "MessageReply"
orderType : “limit_order”. Empty if action is “delete”
orderValidity : “today”. Empty if action is “delete”