Graybox™

Desktop API Specification

Hold Brothers, Inc

Preface

This user manual describes how to use “Graybox Desktop API”. “Graybox Desktop API” is created for Third-party users who want to communicate with through the Interface provided by Graybox for Obtaining Market Data and Order Entry

Jan 2017

CONTENTS

TABLES

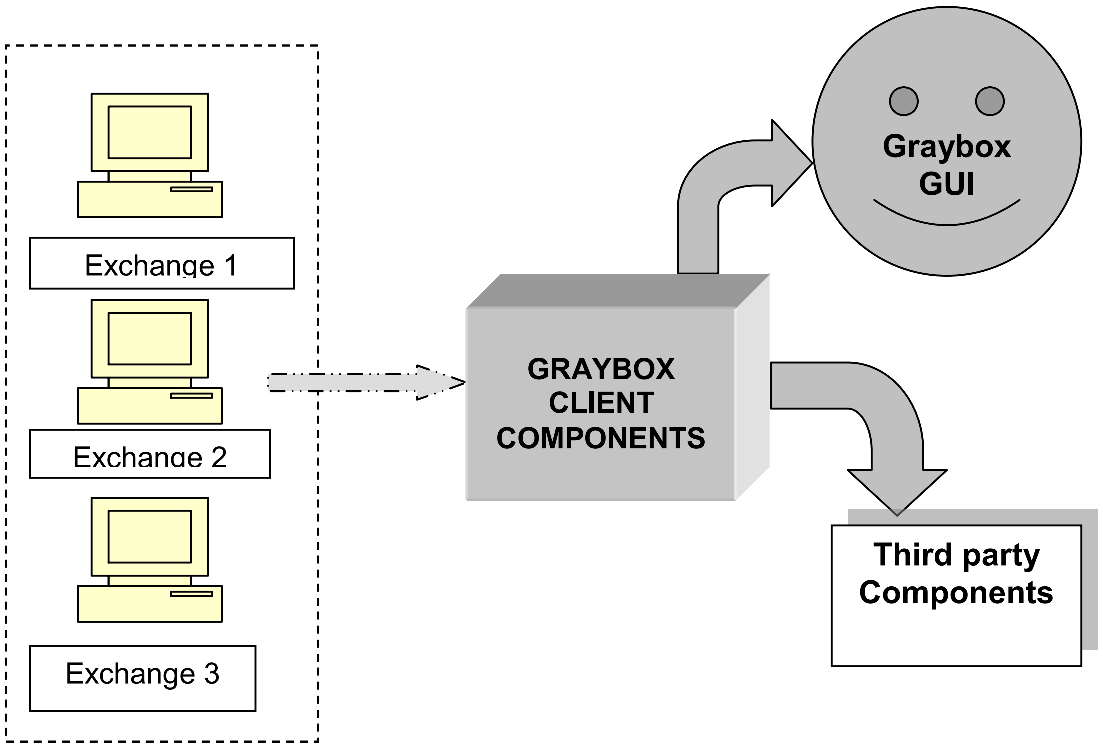
FIGURES

1. About Graybox and Graybox API

1.1. Outline of Graybox

GRAYBOX™ is a Direct Access Trading (DAT) solution for U.S stock market that maximizes trader's ability to perform. This product is designed for speed, stability especially keeping in mind the high volume information. It is customizable to multiple styles such as scalping, momentum, swing and position.

Figure 1 Graybox Function Outline



1.2. Outline of Graybox Desktop API

Graybox Desktop API comes with full access to its application programming interface (API) that allows external applications to access the execution functionality of the Graybox™ trading system and use it as a platform to manage order flow. Simple ActiveX controls send and manage orders, maintain positions, and can be easily integrated with a C/C++, C# or Visual Basic application. Utilize canned code of some of Graybox's popular order logic routines, like smart orders. External Applications can control every aspect of an order, from Time In Force to specifying reserve quantities, risk and compliance features. The API is shipped with a sample application and source code that can help in reducing the integration time cycle.

1.2.1. **Desktop Execution API**

Graybox Execution Application Programming Interface (API) is set of Interfaces which establishes connection with Running Graybox Client and Executes Order. The

Connection Established, Order, Execution can be monitored through Graybox UI for confirmation as well

2. How to get Graybox Desktop API

Graybox Desktop API can be obtained from www.support.hold.com/api . This will install Both Com Component and sample applications developed in VB, VC++ and C#

3. Interface Specification

The following section explains the Interface Specification of Graybox API Specification.

3.1. Desktop Execution API

3.1.1. **Interface: GBXCTRLLib:: IBbx**

An interface through we can place orders for a stock through Graybox which can be also be viewed in Active Order window of Graybox, until they are cancelled, expired or executed.

Public Methods

3.1.2. **BbxInitialize**

Used for establishing a connection with the Graybox

```
      HRESULT BbxInitialize(BSTR IP, int iPort, long hBbxWnd)

*pVal
[in] A null terminated string which specifies the socket.

iPort
[in] An integer value that specifies the port

hBbxWnd
[in] A windows handler, passed as long
```

3.1.3. **BbxSend**

Used to send message to the Graybox after the connection is established

```
      HRESULT BbxSend(BSTR Msg, int iSize)

[in] A null terminated string which specifies the message that needs
to passed to server via socket
```

3.1.4. **BbxShutdown**

Used to disconnect from the Graybox

```
      HRESULT BbxShutdown()
```

3.1.5. **SendOrder**

Used to place order in Graybox by passing the required order information

```
      HRESULT SendOrder(BbxMsgId MsgId, int iBbxId, GbxOrderSide
OrdSide, GbxOrderType OrdType, GbxOrderMode OrdMode, BSTR Symbol, int
iQty, int iDispQty, double dPrc, GbxExchngId ExchngId, GbxDestId
DestId, GbxOrderTif OrdTif, int iTifSec);
```

BbxMsgId
[in] An enum value of type BbxMsgId which has been defined in Gbxctrl.idl for the message Id that needs to be sent to graybox.

iBbxId
[in] An integer vlue indicating the id for the order

OrdSide
[in] An enum value of type GbxOrderSide which has been defined in Gbxctrl.idl indicating the Order side.

OrdType
[in] An enum value of type GbxOrderType which has been defined in Gbxctrl.idl indicating the order type.

Symbol
[in] A null terminated string that inicates the stock symbol.

iQty
[in] An integer value that specifies the quantity that be ordered.

iDispQty
[in] An integer value that specifies the reserve quantity.

dPrc
[in] A double value that specifies the spot price.

ExchngId
[in] An enum value of type GbxOrderSide which has been defined in Gbxctrl.idl indicating the exchange.

GbxDestId
[in] An enum value of type GbxOrderSide which has been defined in Gbxctrl.idl indicating the Destination exchange.

OrdTif
[in] An enum value of type of Time In force which has been defined in Gbxctrl.idl indicating on the execution of an order.

iTifSec

[in] An integer value that reflects Time inforce in seconds for the order if applicable.

3.1.6. SubscribeSymbol

Used to subscribe stock symbol with the exchange

```
      HRESULT SubscribeSymbol (BSTR Symbol);
```

Symbol
[in] A null terminated string which refers to stock symbol.

3.1.7. Order

This is used to place order in Graybox by passing the necessary order information.

```
      HRESULT Order(BbxMsgId MsgId, int iBbxId, GbxOrderSide OrdSide,
GbxOrderType OrdType, GbxOrderMode OrdMode, BSTR Symbol, int iQty,
int iDispQty, double dPrc, GbxExchngId ExchngId, GbxDestId
DestId,GbxOrderTif OrdTif, int iTifSec, double dSpotPrice)
```

BbxMsgId
[in] An enum value of type BbxMsgId which has been defined in Gbxctrl.idl for the message Id that needs to be sent to graybox.

iBbxId
[in] An integer value indicating the id for the order

OrdSide
[in] An enum value of type GbxOrderSide which has been defined in Gbxctrl.idl indicating the Order side.

OrdType
[in] An enum value of type GbxOrderType which has been defined in Gbxctrl.idl indicating the order type.

Symbol
[in] A null terminated string that inicates the stock symbol.

iQty
[in] An integer value that specifies the quantity that be ordered.

iDispQty
[in] An integer value that specifies the reserve quantity.

dPrc
[in] A double value that specifies the spot price.

ExchngId
[in] An enum value of type GbxOrderSide which has been defined in Gbxctrl.idl indicating the exchange.

GbxDestId
[in] An enum value of type GbxOrderSide which has been defined in Gbxctrl.idl indicating the Destination exchange.

```
OrdTif
[in] An enum value of type GbxOrderSide which has been defined in
Gbxctrl.idl indicating on the execution of an order.

iTifSec
[in] An integer value indication the execution of order in seconds.

dSpotPrice
[in] A double value indicating the spot price.
```

3.1.8. **Interface: GBXCTRLLib:: IBbxMsg**

Holds the property associated with the order

Properties

MsgId

A numeric value indicates the message type.

BBxId

A long value which holds the Sequence Number generated by the calling application

OrdSide

A long value indicates the order action. E.g., buy or sell

Symbol

A string value holds the stock symbol for the current order.

Quantity

A long value holds the quantity for the current order.

DispQty

A long value. Indicates the display quantity of the order

Price

A double value holds the limit price for stocks.

ExchngId.

A long value holds the exchange id. The ID for each exchange is defined in GbxExchngId data type.

DestId

A long value holds the destination exchange id. The ID for each exchange is defined in GbxDestId data type.

OrdTif

A long value holds time at which the order was placed. Its defined in the GbxOrderTif data type.

TifSec

A long value which holds the time in force in seconds

OrdType

A long value holds order type. Its defined in the GbxOrderType data type.

GbxId

A long value holds the Id generated by GrayBox.

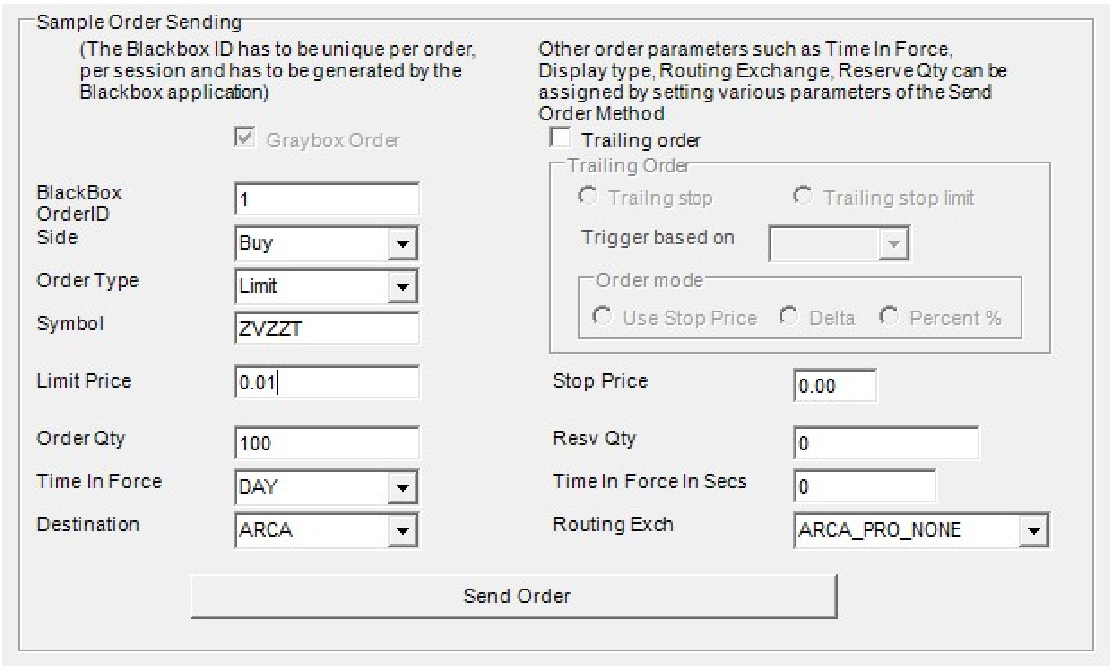
StopPrice

A double value holds the stop price.

3.1.9. **User Interface Order screen**

Graybox Order

Figure 2 Order Entry Screen – Regular order



Sample Order Sending

(The Blackbox ID has to be unique per order, per session and has to be generated by the Blackbox application)

Other order parameters such as Time In Force, Display type, Routing Exchange, Reserve Qty can be assigned by setting various parameters of the Send Order Method

[✓] Graybox Order

[ ] Trailing order

Trailing Order

( ) Trailng stop ( ) Trailing stop limit

Trigger based on

Order mode

( ) Use Stop Price ( ) Delta ( ) Percent %

BlackBox OrderID	1		
Side	Buy		
Order Type	Limit		
Symbol	ZVZZT		
Limit Price	0.01	Stop Price	0.00
Order Qty	100	Resv Qty	0
Time In Force	DAY	Time In Force In Secs	0
Destination	ARCA	Routing Exch	ARCA_PRO_NONE

The above screenshot illustrates how a typical Graybox order is placed through the Graybox API. *Please note BlackBox Order is an auto-increment field that uniquely identifies the order from the Graybox API application.*

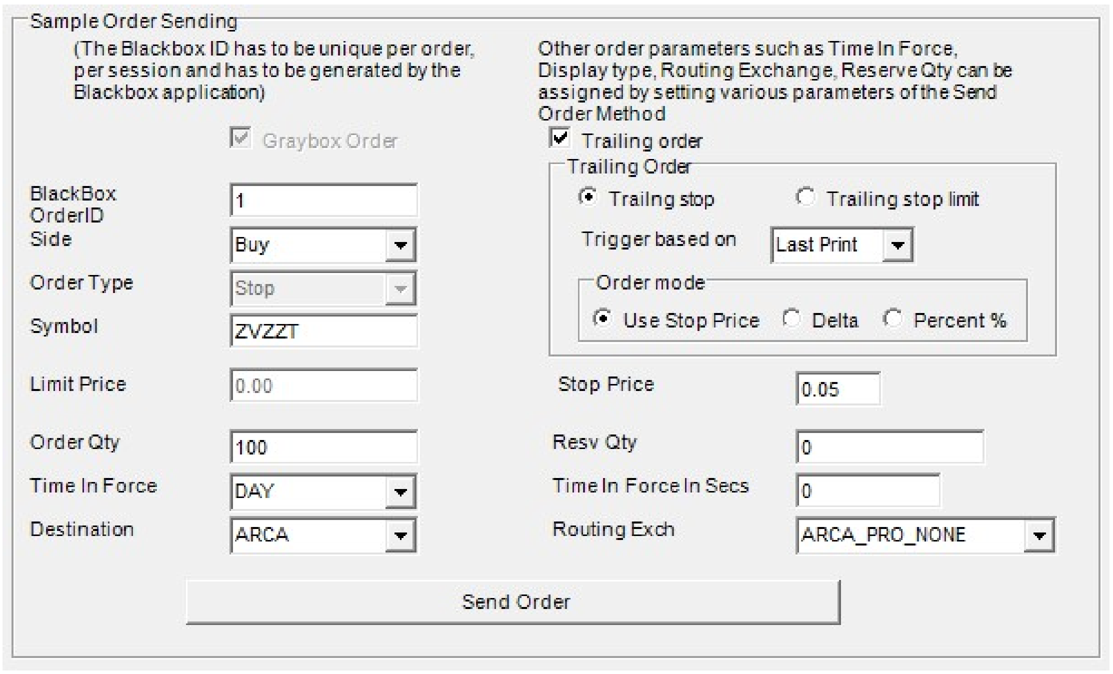
As can be seen, the user fills up the required information like order side, order type, stock symbol, quantity and destination exchange before placing an order. Once the user presses the 'Send Order' button, the API routes the information to the Graybox application.

<u>Sample VB code for sending Graybox order</u>

```
Dim BBtoGB As Bbx
Set BBtoGB = New Bbx

' for Graybox order
BBtoGB.Order MSG_BBX_ORDER, BBID, GBX_BUY, GBX_MARKET, GBX_NORMAL, "MSFT", 100, uiDisplayQty, 0.00, EXCHNG_NYSEPLUS, DEST_ARCA, GBX_USETIFINSECS, cTIFInSecs, 28.88
```

Trailing Order

Figure 3 Order Entry Screen – Trailing stops



Sample Order Sending

(The Blackbox ID has to be unique per order, per session and has to be generated by the Blackbox application)

Other order parameters such as Time In Force, Display type, Routing Exchange, Reserve Qty can be assigned by setting various parameters of the Send Order Method

☑ Graybox Order

☑ Trailing order

Trailing Order

● Trailng stop   ○ Trailing stop limit

Trigger based on: Last Print

Order mode

● Use Stop Price   ○ Delta   ○ Percent %

BlackBox OrderID	1	Stop Price	0.05
Side	Buy	Resv Qty	0
Order Type	Stop	Time In Force In Secs	0
Symbol	ZVZZT	Routing Exch	ARCA_PRO_NONE
Limit Price	0.00		
Order Qty	100		
Time In Force	DAY		
Destination	ARCA		

In case of the Trailing order, the user can either choose Trailing stop (as seen in fig above) or Trailing stop limit. Here, apart from the regular inputs like order side, symbol, quantity and destination exchange, the user has a choice of defining the trailing stop price. This has three variations namely use stop price, delta and percent that would define the actual trailing stop price from the market price for the order. The order can be triggered based on the last print, last ask and last bid. Once the user presses the 'Send Order' button, the API routes the information to the Graybox application.

<u>Sample code for sending Trailing order</u>

```
Dim BBtoGB As Bbx
Set BBtoGB = New Bbx
```

‘ for trailing stop – use stop price
BBtoGB.Order *MSG_BBX_GRAYBOX_STOPS*, BBID, *GBX_STOP*, *GBX_MARKET*, *GBX_GBSTOPS_BASEDON_LASTPRINT*, “MSFT”, 100, uiDisplayQty, 0.00, *EXCHNG_NYSEPLUS*, *DEST_ARCA*, *GBX_USETIFINSECS*, cTIFInSecs, 28.88

‘ for trailing stop – use delta
BBtoGB.Order *MSG_BBX_GRAYBOX_STOPS_TRAILNG_DELTA*, BBID, *GBX_STOP*, *GBX_MARKET*, *GBX_GBSTOPS_BASEDON_LASTPRINT*, “MSFT”, 100, uiDisplayQty, 0.00, *EXCHNG_NYSEPLUS*, *DEST_ARCA*, *GBX_USETIFINSECS*, cTIFInSecs, 28.88

‘ for trailing stop – use percentage
BBtoGB.Order *MSG_BBX_GRAYBOX_STOPS_TRAILNG_DELTA_PCT*, BBID, *GBX_STOP*, *GBX_MARKET*, *GBX_GBSTOPS_BASEDON_LASTPRINT*, “MSFT”, 100, uiDisplayQty, 0.00, *EXCHNG_NYSEPLUS*, *DEST_ARCA*, *GBX_USETIFINSECS*, cTIFInSecs, 28.88

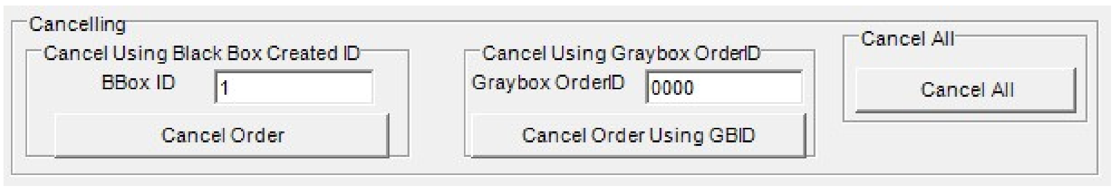
‘ for trailing stop limit – use stop price
BBtoGB.Order *MSG_BBX_GRAYBOX_STOPS*, BBID, *GBX_STOPLIMIT*, *GBX_MARKET*, *GBX_GBSTOPS_BASEDON_LASTPRINT*, “MSFT”, 100, uiDisplayQty, dblLimitPrice, *EXCHNG_NYSEPLUS*, *DEST_ARCA*, *GBX_USETIFINSECS*, cTIFInSecs, 28.88

‘ for trailing stop limit – use delta
BBtoGB.Order *MSG_BBX_GRAYBOX_STOPS_TRAILNG_DELTA*, BBID, *GBX_STOPLIMIT*, *GBX_MARKET*, *GBX_GBSTOPS_BASEDON_LASTPRINT*, “MSFT”, 100, uiDisplayQty, dblLimitPrice, *EXCHNG_NYSEPLUS*, *DEST_ARCA*, *GBX_USETIFINSECS*, cTIFInSecs, 28.88

‘ for trailing stop limit – use percentage
BBtoGB.Order *MSG_BBX_GRAYBOX_STOPS_TRAILNG_DELTA_PCT*, BBID, *GBX_STOPLIMIT*, *GBX_MARKET*, *GBX_GBSTOPS_BASEDON_LASTPRINT*, “MSFT”, 100, uiDisplayQty, dblLimitPrice, *EXCHNG_NYSEPLUS*, *DEST_ARCA*, *GBX_USETIFINSECS*, cTIFInSecs, 28.88

3.1.10. **How to Cancel Order**

Figure 4 Order Cancellation



Here the order cancellation is done using either BBXID or the Graybox ID.

BBXID is the auto-increment numeric value that is used to uniquely tag the order from the Graybox api application. The users needs to merely enter the id of the respective order that has to be cancelled.

Graybox order id is the id that is displayed on the Graybox application’s window for each order placed either through API or the main Graybox application. The user needs to enter this id of what is shown on the Graybox window.

For cancellation the user may choose either of the above ids and click the Cancel Order button.

To cancel all floating order use Cancel All button.

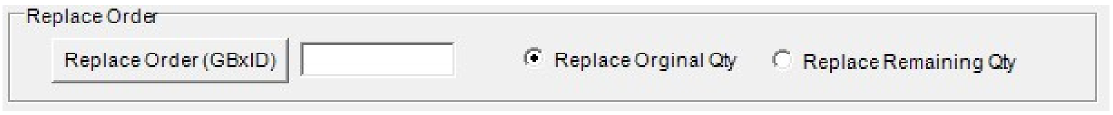
Sample code for cancelling an order

```
‘ for order cancellation using bbxid
BBtoGB.Order ( MSG_BBX_CANCEL, ibbxid, GBX_NOSIDE, GBX_NOTYPE,
GBX_NOMODE, "", 0, 0, 0, EXCHNG_NONE, DEST_NONE, GBX_NOTIF, 0, 0)

‘for order cancellation using GrayboxID
BBtoGB.Order (MSG_BBX_CANCEL, 0, GBX_NOSIDE, GBX_NOTYPE,
GBX_NOMODE, "", 0, 0, 0, EXCHNG_NONE, DEST_NONE, GBX_NOTIF,
iGrayboxid, 0)

‘for cancel all orders
BBtoGB.Order(MSG_BBX_CANCELALL, 0, GBX_NOSIDE, GBX_NOTYPE,
GBX_NOMODE,
"", 0, 0, 0, EXCHNG_NONE, DEST_NONE, GBX_NOTIF, 0, 0)

‘for replace order using Graybox id
BBtoGB.Order(MSG_BBX_REPLACE, 0, CShort(cSide),
CShort(cType_Renamed), ordMode, "", uiQty, 0, dblLimitPrice,
EXCHNG_NONE, DEST_NONE, GBX_NOTIF, iGbxID, 0)
```

3.1.11. How to Replace Order

To replace order which is currently floating user may use Replace Order (GBxID) button. It has the option to replace both original and remaining quntity

Figure 5 Replace Order



Sample code for replacing an order

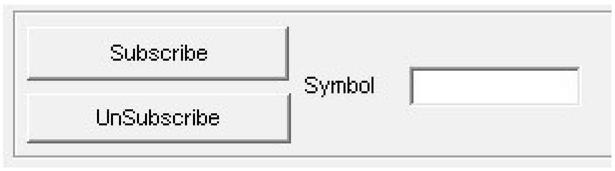
```
‘for replace order using Graybox id – with Original Qty
BBtoGB.Order(MSG_BBX_REPLACE, 0, CShort(cSide),
CShort(cType_Renamed), ordMode, "", uiQty, 0, dblLimitPrice,
GBXCTRLLib.GbxExchngId.REPLACE_SENDING_ORIG_ORDERQTY,
GBXCTRLLib.GbxDestId.DEST_NONE, GBXCTRLLib.GbxOrderTif.GBX_NOTIF,
iGbxID, 0)

‘for replace order using Graybox id – with Remaining Qty
BBtoGB.Order(MSG_BBX_REPLACE, 0, CShort(cSide),
CShort(cType_Renamed), ordMode, "", uiQty, 0, dblLimitPrice,
GBXCTRLLib.GbxExchngId.REPLACE_SENDING_QTYREMAINING,
GBXCTRLLib.GbxDestId.DEST_NONE, GBXCTRLLib.GbxOrderTif.GBX_NOTIF,
iGbxID, 0)
```

3.1.12. How to do a Subscribe or unsubscribe

Graybox will automatically subscribe and unsubscribe to new stocks it encounters during a New Order request. A subscription is done to get compliance and risk related data and for maintain Open PnL on execution.

Optionally, API clients can explicitly subscribe to symbols they are going to trade in before sending an order for it or at the beginning of the session. Its advisable that if the symbol is no longer required, the API clients should explicitly unsubscribe to those symbols. Symbols explicitly subscribed using the API is not automatically unsubscribed when not needed. A large number of symbols subscribed can lead to performance degradation depending upon the nature of the client connectivity to our data center.

Figure 6 Subscribe or unsubscribe



From the sample to subscribe a symbol, just type symbol in the edit box and press subscribe button, to unsubscribe a symbol, just type symbol in the edit box and press Unsubscribe button.

3.1.13. ECN List

Table 1 ECN List

S.No	ECN	ECN(GrayBox)	Constants
1	ALTX	ALTX	DEST_ALTX
2	ARCA	ARCA	DEST_ARCA
3	BATS	BATS	DEST_BATS
4	BATSY	BYXX	DEST_BYXX
5	CSFB	CSFB	DEST_CSFB
6	DTTX	DTTX	DEST_DTTX
7	EBXL	EBXL	DEST_EBXL
8	ECUT	ECUT	DEST_ECUT
9	EDGA	EDGA	DEST_EDGA
10	EDGX	EDGX	DEST_EDGX
11	GBDK	GBDK	DEST_GBDK
12	HBES	NYSE	DEST_NYSE
13	IEXG	IEXG	DEST_ IEXG
15	INET	INET	DEST_ISLD
16	JPMX	JPMX	DEST_JPMX
17	LSTK	LSTK	DEST_LSTK
18	NQPX	NQPX	DEST_NQPX

19	NSDQ	NSDQ	DEST_RASH
20	NSX	NSXS	DEST_NSXS
21	NYFX	NYFX	DEST_NYFX
22	PDQM	PDQM	DEST_PDQM
23	XBOS	XBOS	DEST_XBOS

<u>GBXCTRLLib:: GbxExchngId</u>

This holds the value of the routing exchange.

ALTX

ARCA_ALO_MPL,
ARCA_PL,
ARCA_PNP_BLIND,
ARCA_PNP_LITEONLY,
ARCA_PRO_ACTIVE,
ARCA_PRO_ACTIVE_INSIDELIMT,
ARCA_PRO_ALO,
ARCA_PRO_MPL,
ARCA_PRO_NONE,
ARCA_PRO_NOW,
ARCA_PRO_PSO,
ARCA_PRO_PSOS,

BATS_MPP,
BATS_ONLY,
BATS_POST,
BATS_POSTONLYLMT,
BATS-ROUTE
BATS-MPL-POST-ONLY

BBSS-B3
BBSS-F6
BBSS-C2
BBSS-CE
BBSS-WB
BBSS-WI
BBSS-XR
BBSS-P6
BBSS-LM

BYXX_ONLY,
BYXX_POST,
BYXX_POSTONLYLMT,
BYXX-ROUTE
BYXX-MPP
BYXX-MPL-POST-ONLY

CSFB_CROSS,

DTTX_ROUTING_NITEFAN
DTTX-NITECOVERT
DTTX-SMARTMID
DTTX-PASSIVE
DTTX-TRIM

ECUT

EBXL
EBXL-MID

EDGA_ADDONLY,
EDGA_AGGCROSSLOCK_RDOT,
EDGA_INET,
EDGA_IOCT,
EDGA_IOCX,
EDGA_MPM,
EDGA_ROBA,
EDGA_ROBX,
EDGA_ROIS,
EDGA_ROLF,
EDGA_ROPA,
EDGA_ROUC,
EDGA_ROUD,
EDGA_ROUE,
EDGA_ROUQ,
EDGA_ROUT,
EDGA_ROUX,
EDGA-ROUZ
EDGA-RMPT
EDGA-MPM-ADD
EDGA-ONLY
EDGA-MDO

EDGX_ADDONLY,
EDGX_INET,
EDGX_IOCT,
EDGX_IOCX,
EDGX_MPM,
EDGX_ROBA,
EDGX_ROBX,
EDGX_ROIS,
EDGX_ROLF,
EDGX_ROPA,
EDGX_ROUC,
EDGX_ROUD,
EDGX_ROUE,
EDGX_ROUQ,
EDGX_ROUT,
EDGX_ROUX,
EDGX-ROUZ

EDGX-RDOT
EDGX-ONLY
EDGX-MPM-ADD

EXCHNG_AMEX = 65,
EXCHNG_DEFAULT = 0,
EXCHNG_NONE = 32,
EXCHNG_NYSEDOT = 78,
EXCHNG_NYSEPLUS = 43,

FREE-X
DARK-X
MID-X
GBDK-AGRO
GBDK-PLUS
GBDK-MEDO
GBDK-LOCO

INET
INET-HIDE

JPMX-MID
JPMX-BDARK
JPMX-SMRT
JPMX-POST

NQPX-ADD
NQPX-REMOVE
NQPX-MID

NSDQ_DATA,
NSDQ_DOTA,
NSDQ_DOTI,
NSDQ_DOTM,
NSDQ_DOTN,
NSDQ_DOTP,
NSDQ_MOPP,
NSDQ_NONE,
NSDQ_SCAN,
NSDQ_SKIP,
NSDQ_SPDY,
NSDQ_STGY,
NSDQ_SWIM,
NSDQ-MIDPOINT
NSDQ-MID-ADD

NSXS_ROUTING_DEFAULT,

NYSE-DNS
NYSE-MID
NYSE-MID-ALO

PDQM_ROUTING_ADARK,
PDQM_ROUTING_BATSR,
PDQM_ROUTING_DARKVPS,
PDQM_ROUTING_MID,
PDQM_ROUTING_MIDVPS,
PDQM_ROUTING_PPD,
PDQM_ROUTING_PPOWER,
PDQM_ROUTING_SCAN,
PDQM-DP

XBOS_ADD,
XBOS_REMOVE,
XBOS-MID
XBOS-MID-ADD

GBXCTRLLib:: GbxOrderMode

Holds the value for trailing stop to be calculated based on last ask, last bid, last print

GBX_GBSTOPS_BASEDON_LASTASK
GBX_GBSTOPS_BASEDON_LASTBID
GBX_GBSTOPS_BASEDON_LASTPRINT
GBX_GBSTOPS_BASEDON_LASTPRINT_PRIMARY

GBXCTRLLib:: GbxOrderType

Holds the value for kind of order placed

GBX_LIMIT
GBX_LMTONCLOSE
GBX_MARKET
GBX_MKTONCLOSE
GBX_NOTYPE
GBX_STOP
GBX_STOPLIMIT

GBXCTRLLib:: GbxOrderSide

Holds the value for nature of trade namely BUY, SELL, SHORT

GBX_BUY
GBX_NOSIDE
GBX_SELL
GBX_SHORT

GBXCTRLLib:: GbxOrderTif

Holds the value for Time in force

GBX_ATTHECLOSE
GBX_DAY

GBX_FOK
GBX_GTC
GBX_GTX
GBX_IOC
GBX_NOTIF
GBX_OPN
GBX_USETIFINSECS

GBXCTRLLib:: BbxMsgId

Holds the value for regular orders, trailing stops and get position and others

```
MSG_BBX_ORDER
MSG_BBX_CANCEL
MSG_BBX_REPLACE
MSG_BBX_SMARTORDER
MSG_BBX_GETPOSITION
MSG_BBX_SUBSCRSYMBOL
MSG_BBX_SENDSUPERECN
MSG_BBX_SETPREFWINDOW
MSG_BBX_SETSYMBOLMMWINDOW
MSG_BBX_CANCELALL
MSG_BBX_GRAYBOX_STOPS_TRAILNG_DELTA
MSG_BBX_GRAYBOX_STOPS_TRAILNG_DELTA_PCT
MSG_BBX_GRAYBOX_STOPS
MSG_BBX_UNSUBSCRSYMBOL
MSG_BBX_GETOPENORDERS
MSG_BBX_GETENTITLEMENTS
MSG_BBX_RESET_ALL_GRAYBOX_CONNECTIONS
```

Note: the one in the orange color is supported but not available in sample for now.

3.1.14. How to Process Incoming/Return messages

Graybox interface will fill the structure after processing the order. It is part of `GBXCTRLLib::IBbxMsg`. It holds the following Information

```
MSG_GBX_ORDERCONFIRM          - Order confirmed
MSG_GBX_CANCELCONFIRM         - Cancel confirmed
MSG_GBX_REPLACECONFIRM        - Replace Confirmed
MSG_GBX_ORDEREXEC             - Order Executed
MSG_GBX_CANCELEXEC            - Order Executed
MSG_GBX_CANCELEXEC            - Order Cancelled for ID
MSG_GBX_ORDERERROR            - Order Rejected
MSG_GBX_CANCELERROR           - Cancel Failed
MSG_GBX_REPLACEEXEC           - Order Replaced
MSG_GBX_REPLACEERROR          - Replace failed
MSG_GBX_POSITION              - Position for a Symbol/s
MSG_GBX_ALL_OPEN_ORDERS       - Get All Open Orders
MSG_GBX_CURRENTENTITLEMENTS   - Get Entitlements
MSG_GBX_SYSTEMSTATUS          - status Router, market
MSG_GBX_CONNECTEDTOGRAYBOX    - Gray box Connection status
MSG_GBX_DISCONNECTEDFROMGRAYBOX -Disconnection Status
MSG_GBX_ERRORCONNECTINGTOGRAYBOX- Error in connecting
```

Refer samples for the implementation. The following are the pointers

VB.net : Refer GbxSink_vbxevent function frmmain.vb

VC++ :	Refer Notify function in GBXclnview.cpp
C#	:	Refer GBxEventhandler function in Grayboxapi.cs

Use `IBbxMsg::GetErrorMessage()` to check errors. E.g Reason for rejection etc

4. How to Integrate

4.1. Desktop Execution API

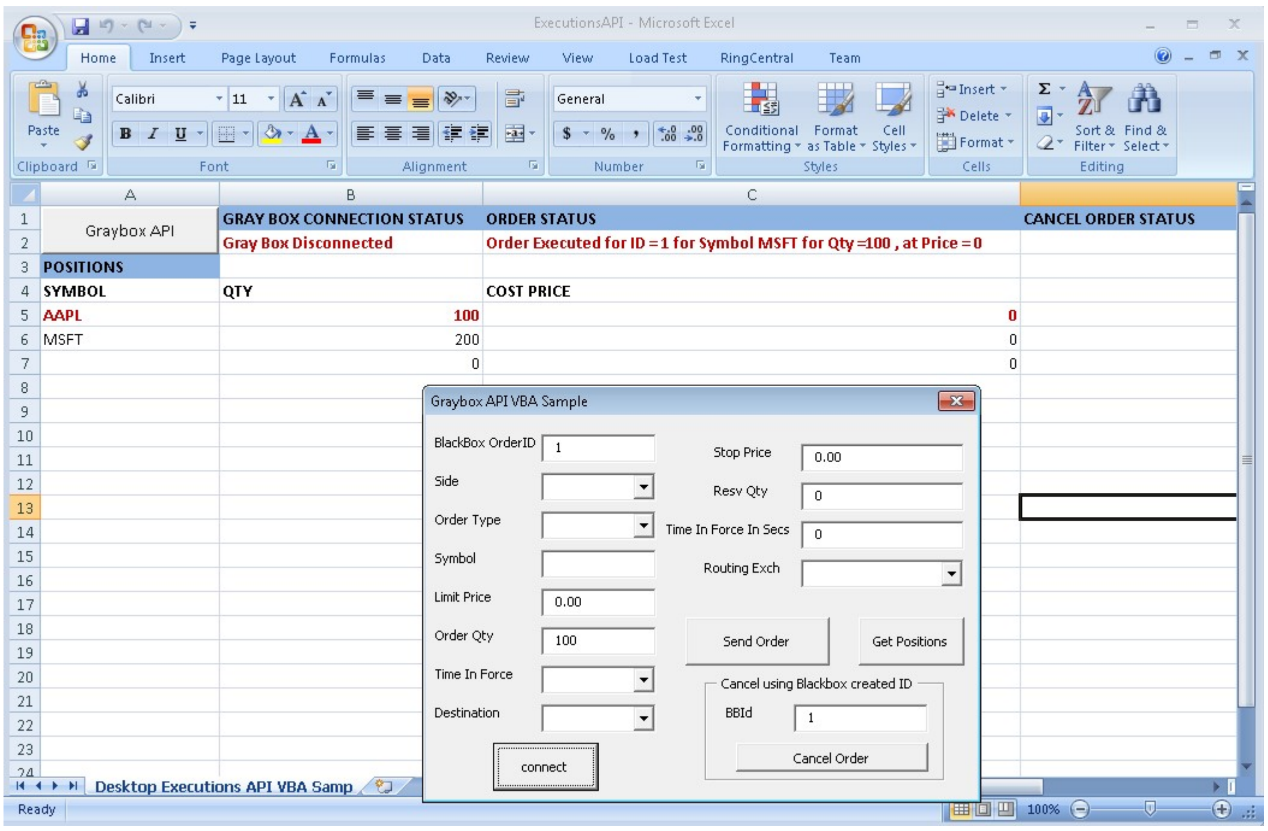
Refer the sample application come with this tool which will demonstrate the Implementation using C# or VB.

5. Excel/VBA User Interface

5.1. Execution API

ExecutionsAPI.xlsm:This document enables us to connect to a running instance of ‘Graybox’ application and send, cancel an order. It also gets position information and populates the excel sheet with all open positions.

Figure 7 Execution API Excel sample



A	B	C	D
Graybox API	GRAY BOX CONNECTION STATUS	ORDER STATUS	CANCEL ORDER STATUS
	Gray Box Disconnected	Order Executed for ID =1 for Symbol MSFT for Qty =100 , at Price =0	
POSITIONS			
SYMBOL	QTY	COST PRICE	
AAPL	100	0	
MSFT	200	0	
	0	0	

Graybox API VBA Sample

BlackBox OrderID	1	Stop Price	0.00
Side		Resv Qty	0
Order Type		Time In Force In Secs	0
Symbol		Routing Exch	
Limit Price	0.00		
Order Qty	100		
Time In Force			
Destination			

Cancel using Blackbox created ID
BBId: 1

ExecutionsAPI - Microsoft Excel
Desktop Executions API VBA Samp
Send Order
Get Positions
Cancel Order
connect
Ready

- This sample needs to connect to a running instance of ‘Graybox’ application

- To send out an order fill in side, order type, symbol, list price , qty, Time In Force, Destination and Stop price respectively and click 'Send Order'.
- Every order is associated with a Blackbox Order ID. The Blackbox order ID text field is self populated and auto incremented. Use this ID in the BBid text field to cancel the order.
- The 'Get Positions' button gets the list of all open positions.
- All status of the sent order is displayed in Cell 'C2' of the sheet.
- All status of the cancelled order is displayed in cell 'D2'.
- All position information is displayed in columns 'A, B and C' respectively' from row 5.
- To get started click the Graybox API button.

The API will support Windows plat form above Windows 2000. the detail is given below

Table 2 Operating System Support Table

The Following OS will be supported

O.S	Supporting Files	Remarks
WindowsXP,2008, Windows 10	All files come with installation	
Windows 7,	All files come with installation	DEP need to be disabled

To Disable DEP – (Vista/Win 7 Specific specific)

A) In the command prompt, type in below and press **Enter**

```
bcdedit.exe /set {current} nx AlwaysOff
```

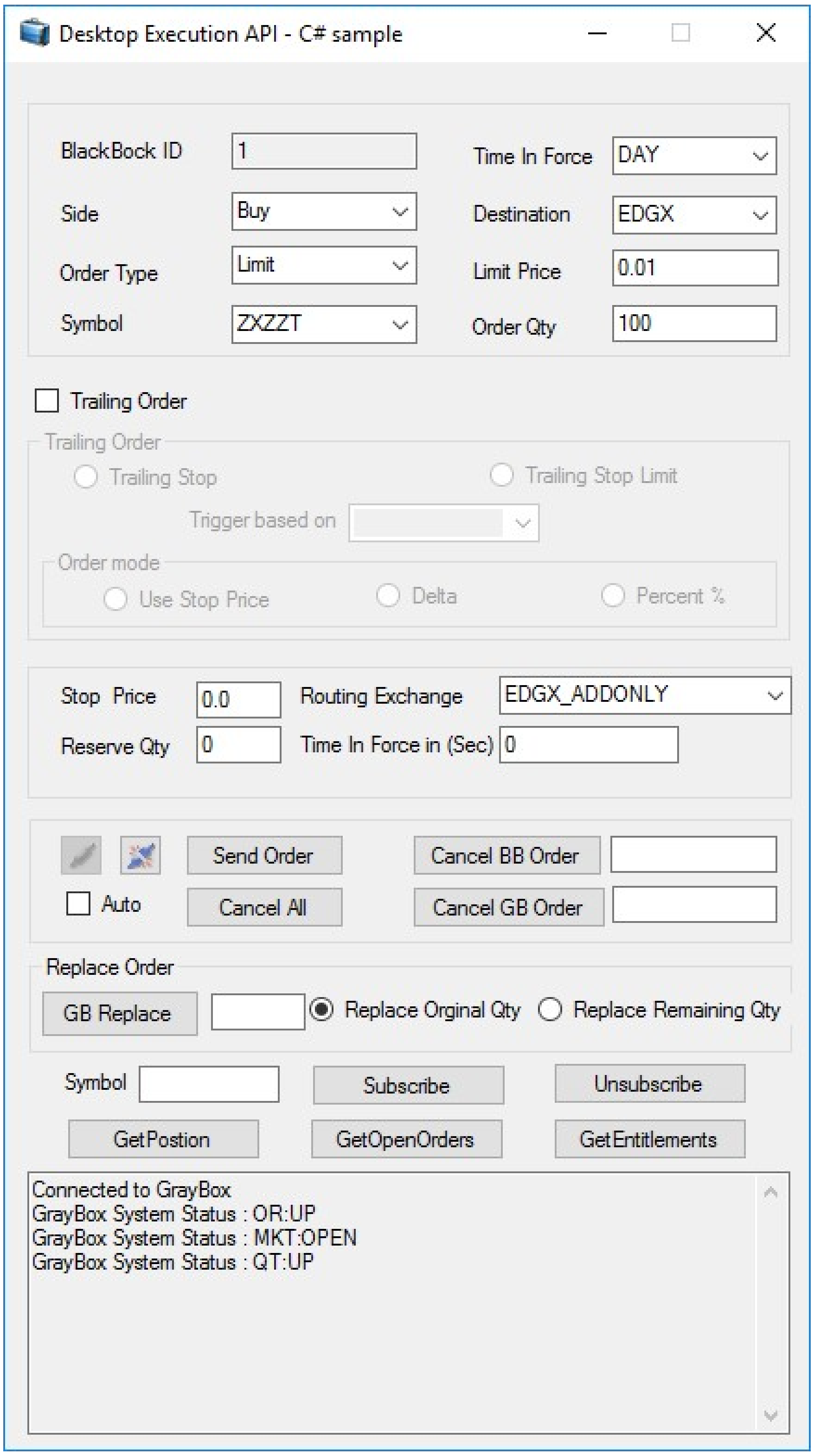
B) You should get a success message back.

C) Close the command prompt.

D) **Restart the computer** to apply.

Alternatively you can also disable for the specific exe using editbin utility which comes with Visual studio

6. Appendix A User Interface's


Desktop Execution API - C# sample

BlackBock ID	1	Time In Force	DAY
Side	Buy	Destination	EDGX
Order Type	Limit	Limit Price	0.01
Symbol	ZXZZT	Order Qty	100

☐ Trailing Order

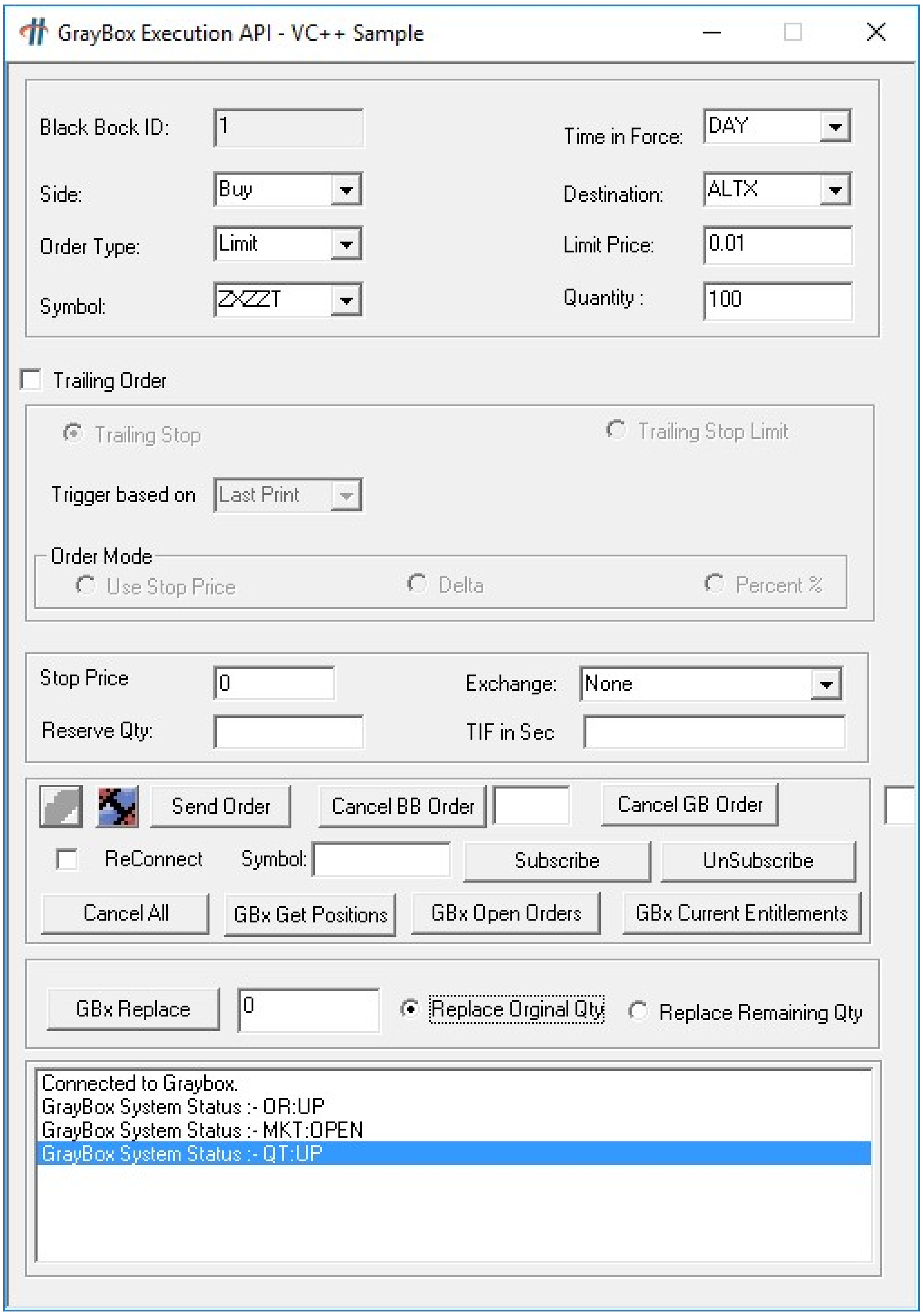
Stop Price	0.0	Routing Exchange	EDGX_ADDONLY
Reserve Qty	0	Time In Force in (Sec)	0

☐ Auto

Replace Order: ● Replace Orginal Qty ○ Replace Remaining Qty

```
Connected to GrayBox
GrayBox System Status : OR:UP
GrayBox System Status : MKT:OPEN
GrayBox System Status : QT:UP
```

Figure 8 C# User Interface
Auto Option will automatically detect Gray box Platform



Black Bock ID: 1
Side: Buy
Order Type: Limit
Symbol: ZXZZT
Time in Force: DAY
Destination: ALTX
Limit Price: 0.01
Quantity : 100

[✓ Trailing Stop] Trailing Stop Limit
Trigger based on Last Print
Order Mode: Use Stop Price / Delta / Percent %

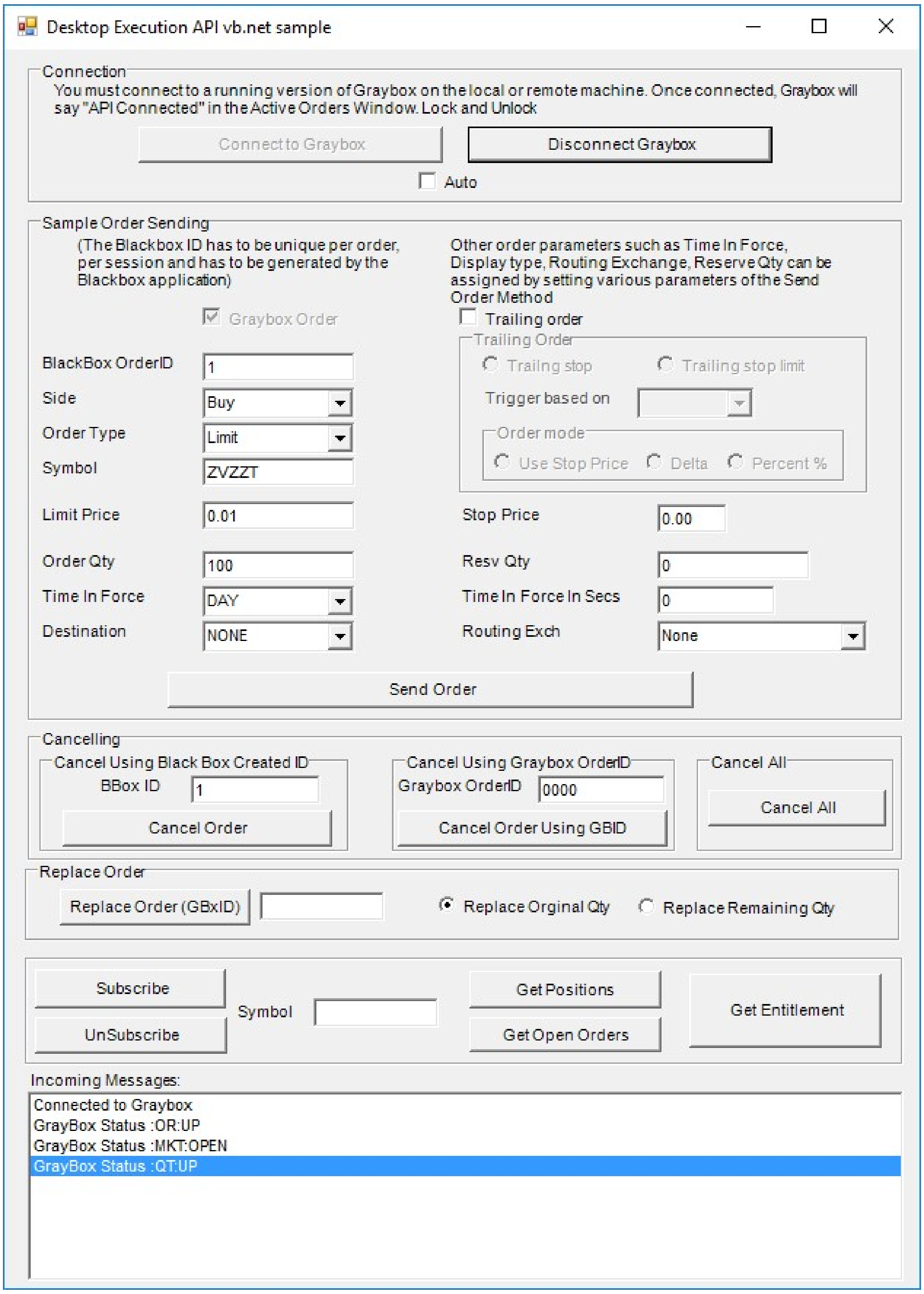
Stop Price 0
Reserve Qty:
Exchange: None
TIF in Sec

Symbol:

0 [✓ Replace Orginal Qty] Replace Remaining Qty

```
Connected to Graybox.
GrayBox System Status :- OR:UP
GrayBox System Status :- MKT:OPEN
GrayBox System Status :- QT:UP
```

Figure 9 VC++ User Interface



Desktop Execution API vb.net sample

Connection

You must connect to a running version of Graybox on the local or remote machine. Once connected, Graybox will say "API Connected" in the Active Orders Window. Lock and Unlock

Connect to Graybox | Disconnect Graybox

☐ Auto

Sample Order Sending

(The Blackbox ID has to be unique per order, per session and has to be generated by the Blackbox application)

Other order parameters such as Time In Force, Display type, Routing Exchange, Reserve Qty can be assigned by setting various parameters of the Send Order Method

☑ Graybox Order

☐ Trailing order

Trailing Order: ○ Trailng stop ○ Trailing stop limit

Trigger based on

Order mode: ○ Use Stop Price ○ Delta ○ Percent %

BlackBox OrderID	1		
Side	Buy		
Order Type	Limit		
Symbol	ZVZZT		
Limit Price	0.01	Stop Price	0.00
Order Qty	100	Resv Qty	0
Time In Force	DAY	Time In Force In Secs	0
Destination	NONE	Routing Exch	None

Send Order

Cancelling

Cancel Using Black Box Created ID: BBox ID 1 — Cancel Order

Cancel Using Graybox OrderID: Graybox OrderID 0000 — Cancel Order Using GBID

Cancel All: Cancel All

Replace Order

Replace Order (GBxID) — ● Replace Orginal Qty ○ Replace Remaining Qty

Subscribe | UnSubscribe | Symbol | Get Positions | Get Open Orders | Get Entitlement

Incoming Messages:

```
Connected to Graybox
GrayBox Status :OR:UP
GrayBox Status :MKT:OPEN
GrayBox Status :QT:UP
```

Figure 10 VB.net User Interface

7. Appendix B Installation Procedure

1. Download the samples and refer gbxctrl.dll from the Graybox installed directory.
2. Graybox should be installed first.

8. Appendix C Contact Information

Send an email to SupportAPI@hold.com Hold team will assist you

____________________End________________