39. CONNECTING TO TRADERAPI

TraderAPI is a .NET .dll that emulates the XTAPI offered by Trading Technologies, Inc., a Chicago-based software company that licenses its X_Trader™ platform to futures traders worldwide. (You can download a free demo version of X_Trader™ along with XTAPI at the company website, www.tradingtechnolgies.com.) Through X_Trader™, we can trade futures contracts on many, many exchanges through one user-friendly graphical user interface. XTAPI, installed as part of X_Trader Pro™, contains COM classes we can instantiate in our .NET programs to, among other functionalites, get real-time market price feeds, route orders, and receive fill notifications. (In a following chapter, I will discuss XTAPI in greater detail.) TraderAPI mimics the look, feel and behavior of this popular tool for electronic market connectivity.

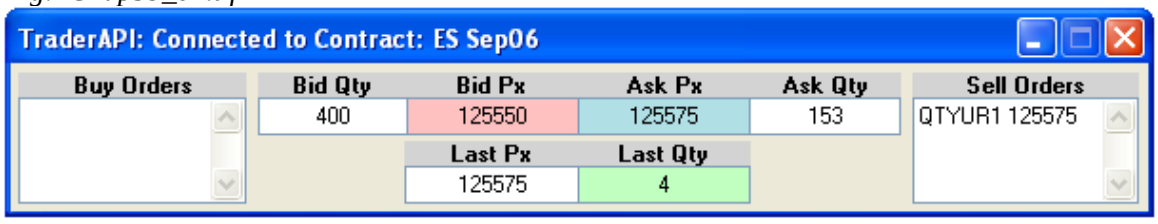
The classes in TraderAPI, desribed in this chapter, are a one-for-one match to real-life classes in XTAPI, though without all of the methods, and the method calls I have included are very similar. However, TraderAPI is not COM as is XTAPI; TraderAPI was written in .NET, so there are differences (which, again will be discussed in a following chapter). Nevertheless, using TraderAPI we can build trading applications with all (or nearly all) the functionality of the real thing.

39.1 TRADERAPI OVERVIEW

TraderAPI has a random, internal price generation mechanism for E-Mini S&P 500, symbol ES, and E-Mini Nasdaq 100, symbol NQ, futures contracts traded on the Chicago Mercantile Exchange. Futher, we can route market and limit orders and receive fill notifications to and from the simulated "exchange."

When a valid connection is made to TraderAPI for a futures contract, a window will appear that will show the exchange view of the instrument including the bid and ask prices, the quantity on the bid and ask, the last trade price and the quantity of the last trade, as well as the order book showing working limit orders we have in the market by order ID number (Key) and price.

Fig: Chap39_01.tif



Now, let's take a look at the TraderAPI classes in alphabetical order.

39.2 FILLOBJ

When a fill is received from the exchange, the OrderSetClass' OnOrderFillData event will fire and pass in information regarding a fill in the form of a FillObj. The get_Get() method will allow us to retrieve this information.

Methods	Description

get_Get(String ^)	Returns BUYSELL, CONTRACT, FFT, KEY, PRICE, PRODUCT, ORDERQTY and/or TIME for a fill.

39.3 INSTROBJCLASS

An instance of the InstrObjClass represents a tradable instrument on the exchange, in this case a futures contract.

Methods	**Description**
get_Get(String ^)	Returns ASK, ASKQTY, BID, BIDQTY, LAST, LASTQTY and/or NETPOS for the instrument
Open(bool)	Opens the InstrObjClass object for updates
Properties	**Description**
CreateNotifyObject	Returns a pointer to a new InstrNotifyClass (observer)
Contract	Gets/sets the contract name (e.g. “Dec06”)
Exchange	Gets/sets the exchange name (e.g. “CME-A”)
OrderSet	Gets/sets OrderSetClass object associated with the instrument
ProdType	Gets/sets the product type (“FUTURE”)
Product	Gets/sets the product (“ES” or “NQ”)
TickSize	Gets the tick size of the contract

39.4 INSTRNOTIFYCLASS

An InstrNotifyClass object is an observer that will raise an event when an update is received from the exchange for its associated InstrObjClass. We can set the filter to receive updates only for specific occurances.

Methods	**Description**
InstrNotifyClass(InstrObjClass)	Constructor
Properties	**Description**
EnablePriceUpdates	Sets ability to receive price updates to true or false
UpdateFilter	Sets filter for update events (e.g. “BID,BIDQTY,ASK,ASKQTY,LAST,LASTQTY”)
Events	**Description**
OnNofityFound	Fires the first time an instrument update is received
OnNotifyUpdate	Fires everytime after the first that an update is received according to the update filter

39.5 ORDEROBJ

An OrderObj represents a working order in the order book (OrderSetClass). Every order is given a unique “Key.”

Methods	**Description**
get_Get(String ^)	Returns ACCT, BUYSELL, ORDERTYPE, PRICE, ORDERQTY, FFT, PRODUCT, CONTRACT, KEY for a working order.
Properties	**Description**

Delete	Cancels and deletes the order

39.6 ORDERPROFILECLASS

An OrderProfileClass object contains all of the information about a trade. To enter an order, we create an OrderProfileClass object and pass it to the SendOrder() method of the OrderSetClass.

Methods	**Description**
get_Get(String ^)	Returns ACCT, BUYSELL, ORDERTYPE, LIMIT, ORDERQTY, FFT, PRODUCT, CONTRACT for the order profile
set_Set(String ^, String ^)	Sets the ACCT, BUYSELL, ORDERTYPE, LIMIT, ORDERQTY, FFT for the order profile
Properties	**Description**
Instrument	Associates an InstrObjClass with the order profile

39.7 ORDERSETCLASS

An OrderSetClass object is our order book and will contain a collection of OrderObj objects. Further, we use an OrderSetClass object to send orders to the exchange, cancel specific working order, delete all working order and receive fill updates from the exchange.

Methods	**Description**
Cancel(String ^)	Cancels an order with a particular key
DeleteOrders(bool)	Deletes all bids (true) and all offers (false)
get_SiteKeyLookup(String ^)	Returns an order with a particular key
Open(bool)	Opens the order set for order entry
SendOrder(OrderProfile ^)	Sends an order
set_Set(String ^, Object ^)	Sets MAXORDERS, MAXORDERQTY, MAXWORKING, MAXPOSITION limits for order entry
Properties	**Description**
EnableOrderAutoDelete	Enables deleting of orders
EnableOrderFillData	Enables fill data evens
EnableOrderSend	Enables order entry
Instrument	Associates an InstrObjClass with the order set
Events	**Description**
OnOrderFillData	Fires when a fill is received
OnOrderSetUpdate	Fires when an update to an order in the order set is received

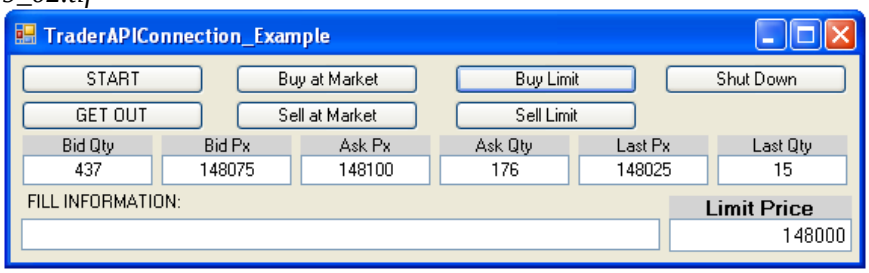
Rather than delve into each method of each object, here is a well-commented program that illustrates most of TraderAPI's functionality.

39.8 CODE SAMPLE: TRADERAPICONNECTION_EXAMPLE

This example will show how to use TraderAPI to get real-time market data, enter orders and receive fill confirmations. In your Windows Forms Application, be sure to add a reference to TraderAPI, create the GUI, and add the appriate code.

When you want to enter a limit order, be sure to enter the price in the text box on the lower right.

Fig: Chap39_02.tif



In this code example, I have left out much error handling code to focus on the topic of TraderAPI.

```
Imports TraderAPI

Public Class Form1

Private WithEvents m_Notify As InstrNotifyClass
Private m_Instr As InstrObjClass
Private WithEvents m_OrderSet As OrderSetClass

Private Sub button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button1.Click

        ' Create a new InstrObjClass object
        m_Instr = New InstrObjClass

        ' Create a new InstrNotifyClass object from the InstrObjClass object.
        m_Notify = m_Instr.CreateNotifyObj
        ' Enable price updates.
        m_Notify.EnablePriceUpdates = True
        ' Set UpdateFilter so event will fire anytime any one of these changes in the
        ' associated InstrObjClass object.
        m_Notify.UpdateFilter = "BIDQTY,BID,ASK,ASKQTY,LAST,LASTQTY"
        ' Set the exchange, product, contract and product type.
        m_Instr.Exchange = "CME"
        m_Instr.Product = "NQ"
        m_Instr.Contract = "Sep05"
        m_Instr.ProdType = "FUTURE"
        ' Open m_Instr.
        m_Instr.Open(True)

        ' Create a new OrderSetClass object.
        m_OrderSet = New OrderSetClass
        ' Set the limits accordingly.  If any of these limits is reached,
        ' trading through the API will be shut down automatically.
        m_OrderSet.set_Set("MAXORDERS", 1000)
        m_OrderSet.set_Set("MAXORDERQTY", 1000)
        m_OrderSet.set_Set("MAXWORKING", 1000)
        m_OrderSet.set_Set("MAXPOSITION", 1000)
'Enable deleting of orders.  Enable the OnOrderFillData event.  Enable order sending.
        m_OrderSet.EnableOrderAutoDelete = True
        m_OrderSet.EnableOrderFillData = True
```

```
        m_OrderSet.EnableOrderSend = True

        ' Open the m_OrderSet.
        m_OrderSet.Open(True)
        ' Associate m_OrderSet with m_Instr.
        m_Instr.OrderSet = m_OrderSet
End Sub

Private Sub OnFill(ByVal pFill As FillObj) Handles m_OrderSet.OnOrderFillData
' Get fill data here with chatty calls.
        textBox7.Text = String.Concat("FILL RECEIVED: ", _
         pFill.get_Get("FFT3"), " ID#: ", _
         pFill.get_Get("KEY"), " Price: ", _
         pFill.get_Get("PRICE"), " B/S: ", _
         pFill.get_Get("BUYSELL"), " QTY: ", _
         pFill.get_Get("QTY"))
End Sub

Private Sub OnUpdate(ByVal pNotify As InstrNotifyClass, ByVal pInstr As
InstrObjClass) Handles m_Notify.OnNotifyUpdate

        ' Get new data from the InstrObjClass using chatty calls here.
        textBox1.Text = pInstr.get_Get("BIDQTY")
        textBox2.Text = pInstr.get_Get("BID")
        textBox3.Text = pInstr.get_Get("ASK")
        textBox4.Text = pInstr.get_Get("ASKQTY")
        textBox5.Text = pInstr.get_Get("LAST")
        textBox6.Text = pInstr.get_Get("LASTQTY")
End Sub


Private Sub button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button2.Click
        SendMarketOrder("B")
End Sub

Private Sub button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button1.Click
        SendMarketOrder("S")
End Sub

Private Sub button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button3.Click
        SendLimitOrder("B", textBox8.Text)
End Sub

Private Sub button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button6.Click
        SendLimitOrder("S", textBox8.Text)
End Sub

Private Sub SendMarketOrder(ByVal m_BS As String)

   ' Create an OrderProfileClass object to contain information about a market order.
        Dim m_Profile As New OrderProfileClass
        m_Profile.Instrument = m_Instr
        m_Profile.set_Set("ACCT", "12345")
```

```
        m_Profile.set_Set("BUYSELL", m_BS)
        m_Profile.set_Set("ORDERTYPE", "M")
        m_Profile.set_Set("ORDERQTY", Convert.ToString(6))
        m_Profile.set_Set("FFT3", "MKT ORDER")

        ' Send the order through m_OrderSet.
        Dim m_Result As Int64 = m_OrderSet.SendOrder(m_Profile)
End Sub

Private Sub SendLimitOrder(ByVal m_BS As String, ByVal m_Px As String)
        ' Send a limit order here.
        Dim m_Profile As New OrderProfileClass
        m_Profile.Instrument = m_Instr
        m_Profile.set_Set("ACCT", "12345")
        m_Profile.set_Set("BUYSELL", m_BS)
        m_Profile.set_Set("ORDERTYPE", "L")
        m_Profile.set_Set("LIMIT", m_Px)
        m_Profile.set_Set("ORDERQTY", Convert.ToString(6))
        m_Profile.set_Set("FFT3", "LMT ORDER")
        Dim m_Result As Int64 = m_OrderSet.SendOrder(m_Profile)
End Sub


Private Sub button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button4.Click
        ' Delete all working buys and sells.
        Dim m_Long As Int64 = m_OrderSet.DeleteOrders(True)
        m_Long = m_OrderSet.DeleteOrders(False)
End Sub

Private Sub button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles button7.Click

        ' Shut down should include explicit object destruction.
        m_Instr = Nothing
        m_OrderSet = Nothing
        m_Notify = Nothing
        GC.Collect()
End Sub
End Class
```

39.9 SUMMARY

In this chapter, we have looked at TraderAPI, an XTAPI emulator, for simulating market connectivity. There are six classes in TraderAPI.dll and we use instances of these classes and their methods to work with real-time data and order execution. In later chapters, we will use TraderAPI to develop automated trading system

Summary

In this chapter, we have looked at two methods for connecting to real markets through connections to the APIs of a popular financial industry software package. Building software for monitoring real-time prices, performing analytics and monitoring trade fills and portfolio risk is absolutely necessary for implementation of an automated trading

system. Several of these key components may already be present in COTS software. APIs allow for proprietary analytics to be built on top of these systems.

You should contact the software provider for full documentation of their API before attempting to build a trading system. The documentation will have all the information on the classes and their functionalities in the API along with sample programs.

Chapter Problems

1.) What is the rule regarding automated order entry of options orders?
2.) What is the problem situation with options order entry we need to resolve?
3.) If the COTS application is a COM object, what would you do?
4.) What is the process for creating objects out of the classes in an API?
5.) Where can you find out more about the objects in an API?

Project One

Project Two