2014

FineTech, Inc

rclarke

Document Version: 2.0

[WIN32, .NET AND COM GATEWAY APIS]

Contents

Introduction

This document describes three order management APIs offered by FineTech that enable traders and systems integrators programmatic access to the FineTech CME Gateway.

We offer three ways to access the gateway functionality (see Figure 1)

1. A managed (.NET) API callable from any .NET language, such as VB.NET, C# or F#
2. A shim DLL that can be called from any language (managed or unmanaged) that supports making Windows API style function calls (C, C++, Delphi, etc.)
3. An ATL based COM object that can be used by older platforms such as VB6 and Delphi or by Microsoft Excel.

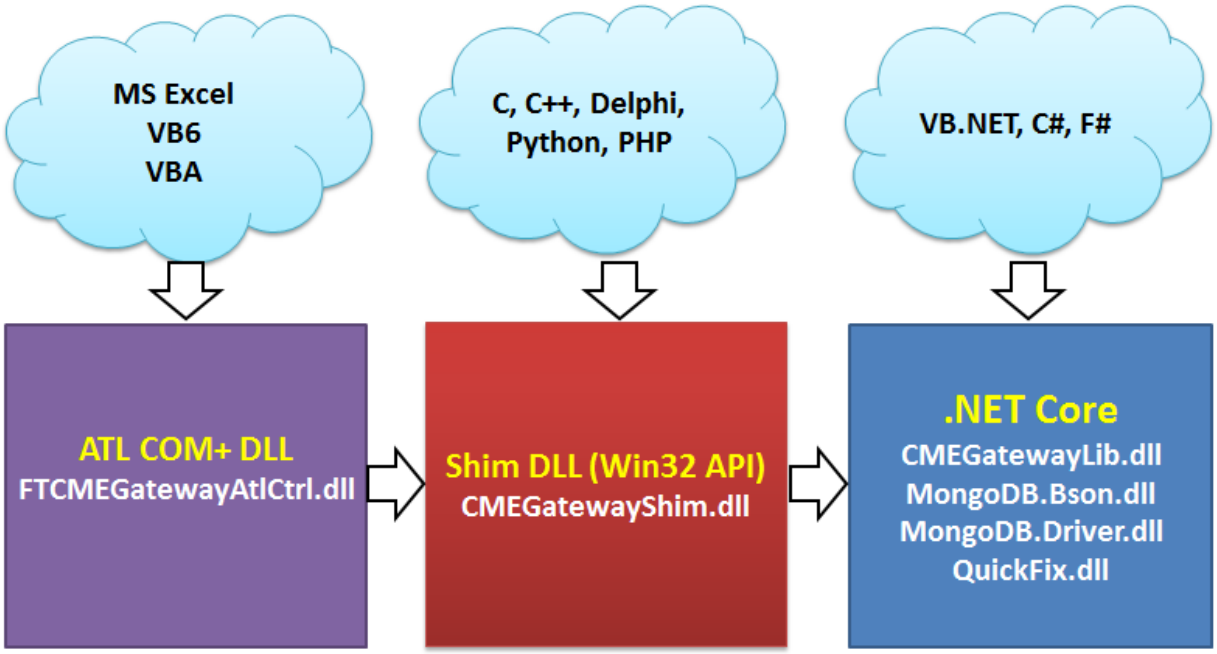


Figure 1. Three ways to access the gateway depending upon your environment

Prerequisites

Two components must be installed before the gateway APIs can be used. If you have Visual Studio 2013, you will have these already. However, client computers almost certainly do not have them and they will need to be installed.

You will need to be logged on as an Administrator in order to carry out this task.

1. Visual C++ 2013 runtime.

   You can download this from here:

   http://www.microsoft.com/en-us/download/details.aspx?id=40784

   Download and run both:

   - vcredist_x64.exe
   - vcredist_x86.exe

2. .NET Framework 4.5

   You can download this from here:

   http://www.microsoft.com/en-us/download/details.aspx?id=30653

   Download and run:

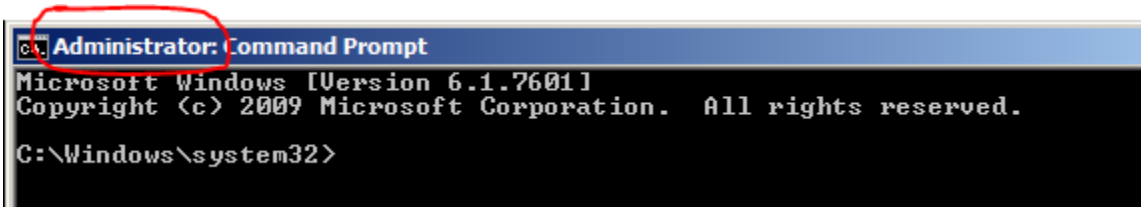
   - dotNetFx45_Full_setup.exe

Installation

The gateway API code base consists of the following files. You can copy these files to any directory you wish on the client computer, for example: `C:\FINETECH`

CMEGatewayLib.dll	Core gateway library (.NET)
MongoDB.Bson.dll	Database library (.NET)
MongoDB.Driver.dll	Database library (.NET)
QuickFix.dll	FIX engine (.NET)
CMEGatewayShim.dll	Win32 Shim
FTCMEGatewayATLCtrl.dll	COM component
FTCMEGatewayATLCtrl.tlb	COM component type library
FTGACTool	Utility to install components in the Global Assembly Cache (client computers that do not have the Windows SDK will not have the GACUTIL command normally used for this purpose)
FinetechExcelDemo.xlsm	Excel sample demonstrating use of the COM component
FineTechCMEAPI.h	Header file for calling CMEGatewayShim.dll

You should install the .NET assemblies in the global assembly cache (GAC). First open up a command window as an administrator.



Navigate to the directory containing the FineTech executables (e.g., C:\FINETECH)

```
CD /D C:\FINETECH

FTGacTool CMEGatewayLib.dll
FTGacTool MongoDB.Bson.dll
FTGacTool MongoDB.Driver.dll
FTGacTool Quickfix.Dll
```

If GACUtil.exe exists, it can be used in place of FTGacTool. FTGacTool is supplied because GACUtil.exe is not likely to exist on client machines that do not have the Windows SDK.

Register the COM component:

```
REGSVR32 FTCMEGatewayATLCtrl.dll
```

Win32 API

Prototypes for the Win32-style API functions are provided in the file `FineTechCMEAPI.h`.

- The functions return **zero** to indicate failure and **non-zero** to indicate success.

- For nearly all of the functions, you pass in a character buffer (inResult) together with the length of this buffer (inResultSize). If the function fails, a reason message will be stored in that buffer. If the message is longer than the buffer you provide, it will be truncated.

Function Name	Description
FTLoadLibrary	`LONG FTLoadLibrary(LPCWSTR pathname)` This function MUST be called first. It loads the CMEGatewayShim.dll library and initializes a set of function pointers for ease of use. Return value: Zero on failure, Non zero on success.
FTInitialize	`LONG FTInitialize(LPSTR inResult, LONG inResultSize, FTInitializeInfo* info)` Initializes the library for use. This function MUST be called after FTLoadLibrary but before any others. **Important: see the description of the FTInitializeInfo structure later in this document**.
FTGetVersion	`LONG FTGetVersion(LPSTR inResult, LONG inResultSize)` Returns a string containing the version number of all the components. Important: You should always call this and store the result somewhere it can be accessed. It is very important we are able to determine accurately which version component is being used, particularly on client sites.
FTStart	`LONG FTStart(LPSTR inResult, LONG inResultSize)` Initiate a connection to the trading server.
FTIsReadyToTrade	`LONG FTIsReadyToTrade()` Returns nonzero if FTInitialize() and FTStart() were successful.

Function Name	Description
FTSendNewOrder	`LONG FTSendNewOrder(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR fix)` Send an order by specifying FIX fields as a delimited string. The delimiter is a pipe sign: | Some fields you can set include: 107 = Security Description (e.g. GCZ4) 55 = Symbol (e.g. GC) 38 = Order Quantity 54 = Side (1: Buy; 2: Sell) 40 = Order Type (1=Market, 2=Limit, 3=Stop, 4=Stop Limit) 44 = Limit Price 99 = Stop Price If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderBuyMarket	`LONG FTSendNewOrderBuyMarket(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty)` Sends a market order to buy 'qty' of instrument 'securityDesc'. If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderSellMarket	`LONG FTSendNewOrderSellMarket(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty)` Sends a market order to sell 'qty' of instrument 'securityDesc'. If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderBuyLimit	`LONG FTSendNewOrderBuyLimit(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty, LPCSTR price)` Sends a limit order to buy 'qty' of instrument 'securityDesc' at the specified 'price' If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderSellLimit	`LONG FTSendNewOrderSellLimit(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty, LPCSTR price)` Sends a limit order to sell 'qty' of instrument 'securityDesc' at the specified 'price'

Function Name	Description
	If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderBuyStop	`LONG FTSendNewOrderBuyStop(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty, LPCSTR price)` Sends a stop order to buy 'qty' of instrument 'securityDesc' at the specified 'price' If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderSellStop	`LONG FTSendNewOrderSellStop(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty, LPCSTR price)` Sends a stop order to sell 'qty' of instrument 'securityDesc' at the specified 'price' If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderBuyStopLimit	`LONG FTSendNewOrderBuyStopLimit(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty, LPCSTR price, LPCSTR stopPx)` Sends a stop/limit order to buy 'qty' of instrument 'securityDesc' at the specified 'price' with a stop price of 'stopPx' If the order is successfully sent, the order number is returned in the id parameter.
FTSendNewOrderSellStopLimit	`LONG FTSendNewOrderSellStopLimit(LPSTR inResult, LONG inResultSize, __int64* id, LPCSTR securityDesc, int qty, LPCSTR price, LPCSTR stopPx)` Sends a stop/limit order to sell 'qty' of instrument 'securityDesc' at the specified 'price' with a stop price of 'stopPx' If the order is successfully sent, the order number is returned in the id parameter.
FTCancelOrder	`LONG FTCancelOrder(LPSTR inResult, LONG inResultSize, __int64 id)` Cancels the order specified in the 'id' parameter. Note that cancel is a request. A non-zero result from this function simply means that the cancel *request* was successfully sent to the trade server. It does not guarantee that the order is canceled.

Function Name	Description
FTCancelReplaceOrder	`LONG FTCancelReplaceOrder(LPSTR inResult, LONG inResultSize, __int64 id, __int64 *newId, int newQty, LPCSTR newPrice, LPCSTR newStopPx)` Sends a request to cancel the order specified in the 'id' parameter and replace it with a new order of quantity 'newQty', limit price 'newPrice' and stop price 'newStopPx'. If 'newQty' is zero, the quantity of the original order is not changed. If 'newPrice' is an empty string, the limit price of the original order (if any) is not changed. If 'newStopPx' is an empty string, the stop price of the original order (if any) is not changed. If the request was successfully sent, the new order Id will be returned in the 'newId' parameter. **Note that cancel replace is a request. A non-zero result from this function simply means that the cancel replace request was successfully sent to the exchange. It does not guarantee that the order has been successfully replaced.**
FTGatewayRequest	`LONG __ FT_GatewayRequest(LPSTR inResult, LONG inResultSize, LPCSTR command, LPCSTR text)` The gateway request allows the client to send various requests to the gateway only (not the exchange) The 'command' parameter defines the type of request The 'text' parameter is not currently used but must be present Valid commands (case sensitive) **Command** / **Purpose** **WO** — Return a list of working orders from the exchange **ML** — Return a moneyline from the gateway See 'gateway response' below for more details.

FTInitializeInfo structure

The FTInitialize API takes a pointer to a control structure used to initialize the library. The structure is defined as follows:

```
typedef struct  {
              LONG apiVersion;
              LPCSTR connectionString;
              fnOnErrorCallback errorCallback;
              fnOnNewCallback newCallback;
              fnOnFillCallback fillCallback;
              fnOnCanceledCallback canceledCallback;
              fnOnConnectionCallback connectionCallback;
              fnOnDisconnectionCallback disconnectionCallback;
              fnOnGatewayResponse gatewayResponseCallback;

              LPCSTR CME_SenderSubID;            // Mandatory Tag 50 (in header)
              LPCSTR CME_SenderLocationID;       // Mandatory Tag 142 (in header)
              LPCSTR CME_TargetSubID;            // Mandatory Tag 57 (in header)

              LPCSTR CME_Account;               // Mandatory Tag 1
              int    CME_CustomerOrFirm;         // Mandatory Tag 204
              LPCSTR CME_ManualOrderIndicator;   // Mandatory Tag 1028
              LPCSTR CME_CtiCode;                // Mandatory Tag 9702

              LPCSTR CME_SelfMatchPreventionID;  // Optional Tag 7928
              LPCSTR CME_InflightMitigation;     // Optional Tag 9768
} FTInitializeInfo;
```

You must initialize <u>every</u> member of this structure prior to calling FTInitialize().

Function Name	Description
apiVersion	Must be set to 1.
connectionString	Trade server connection credentials, a string delimited by semi colons, for example: IP=169.1.1.1;Port=6000;Username=abc1;Password=secret;Encrypted=N IP – the IP address of the trade server Port – the port on the trade server Username = Trading username Password = Trading password Encrypted = Y or N
errorCallback	A user-supplied callback function that the library invokes to notify an error. The function must have the signature: `void onErrorCallback(__int64 id, LPCSTR text)`

Function Name	Description
	id – if the error relates to an order, the order number text – description of the error
newCallback	A user-supplied callback function that the library invokes to notify that a new order has been acknowledged by the trade server and is now in the order book The function must have the signature: `void onNewCallback(__int64 id, LPCSTR symbol, LPCSTR securityDesc, int orderQty)` id – the order id symbol - the instrument symbol (e.g GC) securityDesc – the security description (e.g. GCZ4) orderQty – the quantity open
fillCallback	A user-supplied callback function that the library invokes to notify that a fill has been received on an order The function must have the signature: `void onFillCallback(__int64 id, LPCSTR symbol, LPCSTR securityDesc, int lastShares, LPCSTR lastPx, int orderQty, int cumQty, int leavesQty)` id – the order id symbol - the instrument symbol (e.g GC) securityDesc – the security description (e.g. GCZ4) lastShares – the amount filled in this event lastPx – the price at which the amount was filled orderQty – the quantity on the original order cumQty – the amount filled so far leavesQty – the open quantity still remaining. If zero order is no longer working.
canceledCallback	A user-supplied callback function that the library invokes to notify that an order is no longer working The function must have the signature: `void onCanceledCallback(__int64 id, LPCSTR symbol, LPCSTR securityDesc)` id – the order id

Function Name	Description
	symbol - the instrument symbol (e.g GC) securityDesc – the security description (e.g. GCZ4)
connectionCallback	A user-supplied callback function that the library invokes to notify that the connection to the trade server has been established. The function must have the signature: `void onConnectionCallback()`
disconnectionCallback	A user-supplied callback function that the library invokes to notify that the connection to the trade server has been lost. The function must have the signature: `void onDisconnectionCallback()`
gatewayResponseCallback	A user-supplied callback function that the library invokes when a gateway response messaege is received The function must have this signature: `void onGatewayResponseCallback(LPCSTR command, LPCSTR text)` See 'gateway response' below for further details.

FTInitializeInfo CME Parameters

IMPORTANT.

The CME requires GLOBEX users to specify a number of identifying tags when sending orders.

The Finetech gateway validates that these are present, however it does not validate that their content is correct.

It is the trader's responsibility to ensure the tags are set correctly.

Mandatory tags are:

FTInitializeInfo Member	CME Tag Number
CME_SenderSubID	50
CME_SenderLocationID	142
CME_TargetSubID	57
CME_Account	1
CME_CustomerOrFirm	204
CME_ManualOrderIndicator	1028
CME_CtiCode	9702

Optional tags are:

FTInitializeInfo Member	CME Tag Number
CME_SelfMatchPreventionID	7928
CME_InflightMitigation	9768

For more details on the meaning of these tags, please refer to
http://www.cmegroup.com/confluence/display/EPICSANDBOX/New+Order

Gateway Response

The FTGatewayRequest function allows clients to query various aspects of the gateway.
Currently it is possible to query for working orders (WO) or moneyline (ML).

As part of the API initialization you provide a callback function that is invoked with two parameters: command and text.
The command is the same as the command you send in the gateway request (WO or ML).
The text is a comma-separated list of key value pairs.

WO - Working Orders Response

If you send a gateway request with command=WO the gateway responds with a list of working orders.

The first response tells you how many working orders there are, and provides a report ID, e.g:

```
ReportID=7030251430003,OrderCount=2
```

If OrderCount is zero, there will be no further responses. In this example there will be two further responses. Each will have the same ReportID.

```
ReportID=7030251430003,Entry=1,ClOrdID=7030950500000,OrderID=76132083809,SecurityDesc=GCN4,Symbol=GC,SecurityType=FUT,OrderQty=10,CumQty=0,LeavesQty=10,Price=12000,StopPx=,OrdType=LMT,TimeInForce=Day,Side=Buy
```

```
ReportID=7030251430003,Entry=2,ClOrdID=703095918100001,OrderID=76132084709,SecurityDesc=GCN4,Symbol=GC,SecurityType=FUT,OrderQty=10,CumQty=0,LeavesQty=10,Price=12000,StopPx=,OrdType=LMT,TimeInForce=Day,Side=Buy
```

IMPORTANT. The ability to query working orders is intended to be used to initialize a GUI or as a check that no orders remain working when a trading platform is about to be shut down.

DO NOT write strategies that constantly poll the working orders. This is incorrect anyway, because a working order may be filled or canceled immediately after the report is sent, rendering its content useless for decision making. CME will regard polling for working orders as an abuse of the platform and will disable the session.

ML - Money Line Response

If you send a gateway request with command=ML the gateway responds with a money line.

```
AccountID=1,Balance=100000,OpenPNL=0,ClosedPNL=0,MarginRequired=1000,MarginUtilization=1,TotalFunds=100000
```

All amounts are in USD except MarginUtilization which is a percentage

Win32 Sample

```
#include <stdio.h>
#include <tchar.h>
#include <windows.h>
#include <assert.h>
#include <string>
#include <iostream>
#include "FineTechCMEAPI.h"

void onErrorCallback(__int64 id, LPCSTR text)
{
	std::cout << "Error! id=" << id << ", text=" << text << std::endl;
}

void onNewCallback(__int64 id, LPCSTR symbol, LPCSTR securityDesc, int orderQty)
{
	std::cout << "New Order! id=" << id << ", symbol=" << symbol << ", securityDesc=" <<
securityDesc << ", orderQty=" << orderQty << std::endl;
}

void onFillCallback(__int64 id, LPCSTR symbol, LPCSTR securityDesc, int lastShares, LPCSTR lastPx, int
orderQty, int cumQty, int leavesQty)
{
	std::cout << "Fill! id=" << id << ",symbol=" << symbol << ",securityDesc=" << securityDesc <<
",lastShares=" << lastShares << ",lastPx=" << lastPx << ",orderQty=" << orderQty << ", cumQty=" <<
cumQty << ", leavesQty=" << leavesQty << std::endl;
}

void onCanceledCallback(__int64 id, LPCSTR symbol, LPCSTR securityDesc)
{
	std::cout << "Canceled! id=" << id << ",symbol=" << symbol << ",securityDesc=" << securityDesc
<< std::endl;
}

void onConnectionCallback()
{
	std::cout << "Connected to server" << std::endl;

	__int64 id = 0;
	char result[1000];
	if (FTSendNewOrderBuyLimit(result, sizeof(result), &id, "GCN4", 10, "12000")){
		std::cout << "Order sent, id=" << id << std::endl;
	}
	else {
		std::cout << "FTSendNewOrderBuyLimit failed: " << result << std::endl;
	}

	FTGatewayRequest(result, sizeof(result), "ML", "");
	FTGatewayRequest(result, sizeof(result), "WO", "");
}

void onDisconnectionCallback()
{
	std::cout << "Disconnected from server" << std::endl;
```

```
}

void onGatewayResponseCallback(LPCSTR command, LPCSTR text)
{
	std::cout << "Gateway response: " << command << " : " << text << std::endl;
}

int _tmain(int argc, _TCHAR* argv[])
{
	FTInitializeInfo control;
	memset(&control, 0, sizeof(control));
	control.apiVersion = FT_API_VERSION;
	control.connectionString = "username=AAAA;password=12345;ip=218.145.27.134;port=5000";
	control.canceledCallback = onCanceledCallback;
	control.connectionCallback = onConnectionCallback;
	control.disconnectionCallback = onDisconnectionCallback;
	control.errorCallback = onErrorCallback;
	control.fillCallback = onFillCallback;
	control.newCallback = onNewCallback;
	control.gatewayResponseCallback = onGatewayResponseCallback;

	control.CME_Account = "606060";
	control.CME_CtiCode = "2";
	control.CME_CustomerOrFirm = 0;
	control.CME_InflightMitigation = "N";
	control.CME_ManualOrderIndicator = "N";
	control.CME_SenderLocationID = "US,IL";
	control.CME_SenderSubID = "FTA";

	if (FTLoadLibrary(_T("C:\\finetech\\CMEGatewayShim.dll")))
	{
		char result[1000];
		if (FTInitialize(result, sizeof(result), &control))
		{
			if (!FTStart(result, sizeof(result)))
			{
				std::cout << "FTStart failed: " << result << std::endl;
			}
		}
		else {
			std::cout << "FTInitialize failed: " << result << std::endl;
		}
	}
	else
	{
		std::cout << "Failed to load  library";
	}
	std::cout << "Ready to exit: ";
	_getch();
	return 0;
}
```

Component Object Model (COM) Control

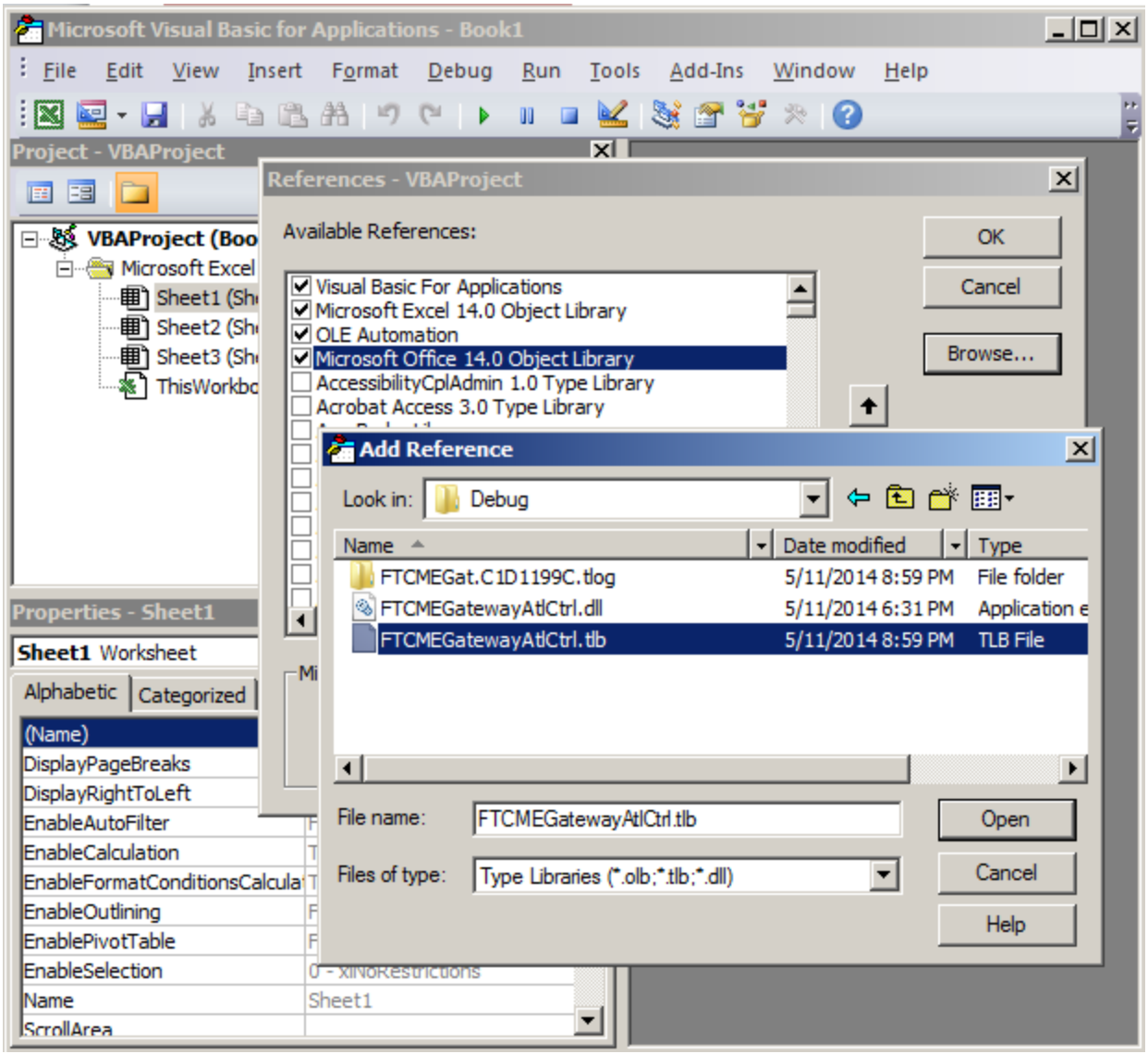
The FTCMEGatewayATLCtrl COM control can be used from any language or tool that is COM-aware, for example, Microsoft Excel.

Method/Event Name	Description
Initialize	`HRESULT Initialize(BSTR shimDLLPath, BSTR connectionString);` This function MUST be called first. `shimDLLPath` – full pathname to the CMEGatewayShim.dll library .
Start	`HRESULT SendNewOrder([in] BSTR fixString, [out, retval] BSTR* orderId);` Start connection to the trade server
GetVersion	`HRESULT GetVersion([out, retval] BSTR* version);` Get the version string of the components. You are strongly recommended to display this where it is easy to see as it is extremely valuable in troubleshooting.
SendNewOrder	`HRESULT SendNewOrder([in] BSTR fixString, [out, retval] BSTR* orderId);` See description of FTSendNewOrder (Win32)
SendNewOrderBuyMarket	`HRESULT SendNewOrderBuyMarket(BSTR securityDesc, int qty, [out, retval] BSTR* orderId);` See description of FTSendNewOrderBuyMarket (Win32)
SendNewOrderSellMarket	`HRESULT SendNewOrderSellMarket([in] BSTR bsSecurityDesc, [in] int qty, [out, retval] BSTR* orderId);` See description of FTSendNewOrderSellMarket (Win32)
SendNewOrderBuyLimit	`HRESULT SendNewOrderBuyLimit([in] BSTR bsSecurityDesc, [in] int qty, [in] BSTR price, [out, retval] BSTR* orderId);` See description of FTSendNewOrderBuyLimit (Win32)

Method/Event Name	Description
SendNewOrderSellLimit	HRESULT SendNewOrderSellLimit([in] BSTR bsSecurityDesc, int qty, [in] BSTR price, [out, retval] BSTR* orderId); See description of FTSendNewOrderSellLimit (Win32)
SendNewOrderBuyStop	HRESULT SendNewOrderBuyStop([in] BSTR bsSecurityDesc, [in] int qty, [in] BSTR price, [out, retval] BSTR* orderId); See description of FTSendNewOrderBuyStop (Win32)
SendNewOrderSellStop	HRESULT SendNewOrderSellStop([in] BSTR bsSecurityDesc, [in] int qty, [in] BSTR price, [out, retval] BSTR* orderId); See description of FTSendNewOrderSellStop (Win32)
SendNewOrderBuyStopLimit	HRESULT SendNewOrderBuyStopLimit([in] BSTR bsSecurityDesc, [in] int qty, [in] BSTR price, [in] BSTR stopPx, [out, retval] BSTR* orderId); See description of FTSendNewOrderBuyStopLimit (Win32)
SendNewOrderSellStopLimit	HRESULT SendNewOrderSellStopLimit([in] BSTR bsSecurityDesc, [in] int qty, [in] BSTR price, [in] BSTR stopPx, [out, retval] BSTR* orderId); See description of FTSendNewOrderSellStopLimit (Win32)
CancelOrder	CancelOrder([in] BSTR orderId); See description of FTCancelOrder (Win32)
CancelReplaceOrder	CancelReplaceOrder([in] BSTR orderId, [in] int newQty, [in] BSTR newPrice, [in] BSTR newStopPx, [out, retval] BSTR* newOrderId); See description of FTCancelReplaceOrder (Win32)

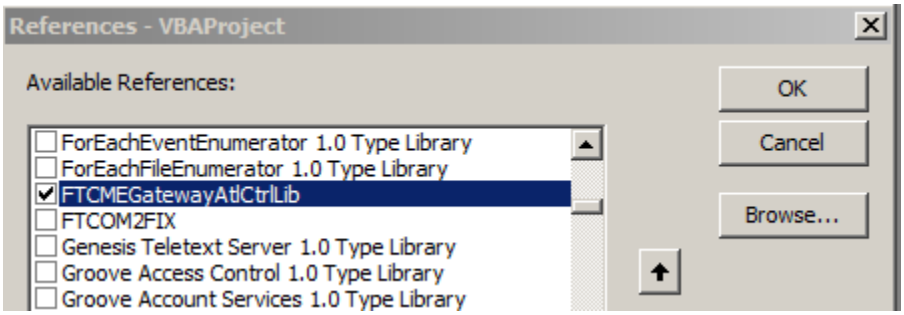
COM Sample

To use the component with Excel, first start Excel. Then enter Alt+F11 to enter VBA mode.



Select Tools>References>Browse and locate FTCMEGatewayAtlCtrl.tlb

Then scroll down, locate “FTCMEGatewayAtlCtrlLib” , check it and click OK:



Because the component raises events, you’ll need to create a class module to access it. You can do that using Insert>Class Module. Here is a sample VBA class module:

```
‘Place the following code in CLASS MODULE:
Option Explicit

Private WithEvents O As CGatewayControl

Private Sub Class_Initialize()
    Set O = New CGatewayControl
End Sub

Public Function GetVersion() As String
    GetVersion = O.GetVersion()
End Function

Public Sub Init()
    Log "Initializing...."  'Specify pathname to shim DLL below:-
    O.Initialize "c:\git\finetech\cmegateway\bin\Debug\cmegatewayshim.dll", _
                 "username=AAAA;password=12345;ip=218.145.27.134;port=5000"
End Sub

Public Sub Connect()
    Log "Connecting..."
    O.Start
End Sub

Public Sub Cancel(id As String)
    If IsNumeric(id) Then
        Log "Send cancel: " & id
        O.CancelOrder id
    End If
End Sub
```

```
Public Sub Replace(id As String, qty As Integer, price As String, stopPx As String)
    Dim newId As String
    newId = O.CancelReplaceOrder(id, qty, price, stopPx)
End Sub

Public Sub BuyMarket(product As String, qty As Integer)
    Log "Buy " & qty & " " & product & " at market (" & O.SendNewOrderBuyMarket(product, qty) &
")"
End Sub

Public Sub SellMarket(product As String, qty As Integer)
    Log "Sell " & qty & " " & product & " at market (" & O.SendNewOrderSellMarket(product, qty) &
")"
End Sub

Public Sub BuyLimit(product As String, qty As Integer, price As String)
    Log "Buy " & qty & " " & product & " at " & price & " (" & O.SendNewOrderBuyLimit(product,
qty, price) & ")"
End Sub

Public Sub SellLimit(product As String, qty As Integer, price As String)
    Log "Sell " & qty & " " & product & " at " & price & " (" & O.SendNewOrderSellLimit(product,
qty, price) & ")"
End Sub

Public Sub BuyStop(product As String, qty As Integer, price As String)
    Log "Buy " & qty & " " & product & " stop at " & price & " (" & O.SendNewOrderBuyStop(product,
qty, price) & ")"
End Sub

Public Sub SellStop(product As String, qty As Integer, price As String)
    Log "Sell " & qty & " " & product & " stop at " & price & " (" &
O.SendNewOrderSellStop(product, qty, price) & ")"
End Sub

Public Sub BuyStopLimit(product As String, qty As Integer, price As String, stopPx As String)
    Log "Buy " & qty & " " & product & " at " & price & " stop " & stopPx & " (" &
O.SendNewOrderBuyStopLimit(product, qty, price, stopPx) & ")"
End Sub

Public Sub SellStopLimit(product As String, qty As Integer, price As String, stopPx As String)
    Log "Sell " & qty & " " & product & " at " & price & " stop " & stopPx & " (" &
O.SendNewOrderSellStopLimit(product, qty, price, stopPx) & ")"
End Sub

Private Sub O_OnCanceled(ByVal id As String, ByVal Symbol As String, ByVal SecurityDesc As String)
    DeleteOrder id
```

```
    Log "Order canceled: " & id & "," & Symbol & "," & SecurityDesc
End Sub

Private Sub O_OnConnection()
    Log "Server connection established"
End Sub

Private Sub O_OnDisconnection()
    Log "Server connection lost"
End Sub

Private Sub O_OnError(ByVal id As String, ByVal ErrorText As String)
    DeleteOrder id
    Log "Error! Order=" & id & " Text=" & ErrorText
End Sub

Private Sub O_OnFill(ByVal id As String, ByVal Symbol As String, ByVal SecurityDesc As String,
ByVal LastShares As Long, ByVal lastPx As String, ByVal OrderQty As Long, ByVal CumQty As Long,
ByVal LeavesQty As Long)
    If LeavesQty = 0 Then DeleteOrder id
    Log "Order Fill: " & id & "," & Symbol & "," & SecurityDesc & ", filled=" & LastShares & ",
price=" & lastPx & ", orderQty=" & OrderQty & ", cumQty=" & CumQty & ", leavesQty=" & LeavesQty
End Sub

Private Sub O_OnNew(ByVal id As String, ByVal Symbol As String, ByVal SecurityDesc As String,
ByVal OrderQty As Long)
    AddOrder id
    Log "Order Ack: " & id & "," & Symbol & "," & SecurityDesc & ", orderQty=" & OrderQty
End Sub
```

The code above relies on some global methods. Create a regular module using Insert>Module:

```
‘Place this code in a regular module
Option Explicit

Global finetechGateway As Gateway

Private Sub clearLog()
    Columns("A:A").ColumnWidth = 100
    Columns("A:A").Select
    Selection.ClearContents
    Range("A1").Select
End Sub

Private Sub clearBook()
    Range("B3").Select
    Range("B3:B999").Select
```

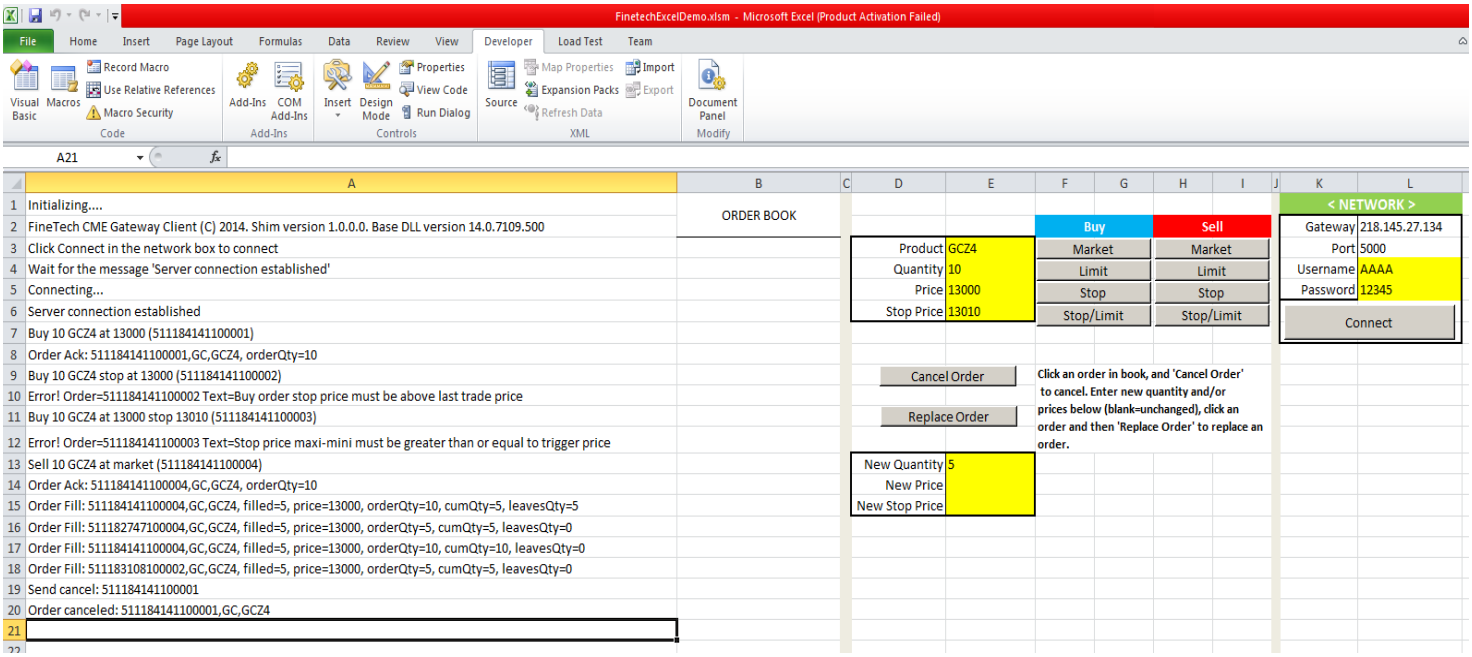
```
    Selection.ClearContents
End Sub

 Public Sub Auto_Open()
    clearLog
    clearBook
    ensureGateway
    Log "Click Connect in the network box to connect"
    Log "Wait for the message 'Server connection established'"
 End Sub

 Private Sub SelectFirstBlankCell(column As Integer, row As Integer)
    Dim rowCount As Integer, currentRow As Integer
    Dim currentRowValue As Variant
    Dim done As Boolean

    rowCount = Cells(Rows.Count, column).End(xlUp).row

    For currentRow = row To rowCount
        currentRowValue = Cells(currentRow, column).Value
        If IsEmpty(currentRowValue) Or currentRowValue = "" Then
            Cells(currentRow, column).Select
            done = True
            Exit For
        End If
    Next

    If Not done Then
        If rowCount < row Then
            Cells(row, column).Select
        Else
            Cells(rowCount + 1, column).Select
        End If
    End If
End Sub

Public Sub AddOrder(id As String)

    SelectFirstBlankCell 2, 3
    Debug.Print "AddOrder, current cell R= " & ActiveCell.row & " C=" & ActiveCell.column

     ActiveCell.FormulaR1C1 = id
End Sub
```

```
Public Sub DeleteOrder(id As String)
    Dim rowCount As Integer, currentRow As Integer
    Dim currentRowValue As Variant
    Dim done As Boolean

    rowCount = Cells(Rows.Count, 2).End(xlUp).row

    For currentRow = 1 To rowCount
        currentRowValue = Cells(currentRow, 2).Value
        If currentRowValue = id Then
            Cells(currentRow, 2).FormulaR1C1 = ""

            Exit For
        End If
    Next
End Sub

Public Sub Log(text As String)
    SelectFirstBlankCell 1, 1

    ActiveCell.FormulaR1C1 = text
End Sub

Sub ensureGateway()
    If finetechGateway Is Nothing Then
        clearLog
        Set finetechGateway = New Gateway

        finetechGateway.Init
        Log finetechGateway.GetVersion()
    End If

End Sub

Sub cancel_Click()
    ensureGateway
    If ActiveCell.column = 2 And IsNumeric(ActiveCell.FormulaR1C1) Then
        finetechGateway.Cancel ActiveCell.FormulaR1C1
    End If
End Sub

Sub replace_Click()
    ensureGateway
    If ActiveCell.column = 2 And IsNumeric(ActiveCell.FormulaR1C1) Then
        finetechGateway.Replace ActiveCell.FormulaR1C1, getNewQuantity(), getNewPrice(),
getNewStopPx()
    End If
End Sub
```

```
Sub connectButton_Click()
    ensureGateway
    finetechGateway.Connect
End Sub

Private Function getProduct() As String
    getProduct = Range("E3")
End Function

Private Function getQuantity() As Integer
    getQuantity = Val(Range("E4"))
End Function

Private Function getPrice() As String
    getPrice = Range("E5")
End Function

Private Function getStopPx() As String
    getStopPx = Range("E6")
End Function

Private Function getNewQuantity() As Integer
    getNewQuantity = Val(Range("E13"))
End Function

Private Function getNewPrice() As String
    getNewPrice = Range("E14")
End Function

Private Function getNewStopPx() As String
    getNewStopPx = Range("E15")
End Function

Sub buyMarketButton_Click()
    ensureGateway
    finetechGateway.BuyMarket getProduct(), getQuantity()
End Sub

Sub buyLimitButton_Click()
    ensureGateway
    finetechGateway.BuyLimit getProduct(), getQuantity(), getPrice()
End Sub

Sub buyStopButton_Click()
    ensureGateway
    finetechGateway.BuyStop getProduct(), getQuantity(), getPrice()
End Sub
```

```
Sub buyStopLimitButton_Click()
    ensureGateway
     finetechGateway.BuyStopLimit getProduct(), getQuantity(), getPrice(), getStopPx()
End Sub

Sub sellMarketButton_Click()
    ensureGateway
    finetechGateway.SellMarket getProduct(), getQuantity()
End Sub

Sub sellLimitButton_Click()
    ensureGateway
      finetechGateway.SellLimit getProduct(), getQuantity(), getPrice()
End Sub

Sub sellStopButton_Click()
    ensureGateway
     finetechGateway.SellStop getProduct(), getQuantity(), getPrice()
End Sub

Sub sellStopLimitButton_Click()
    ensureGateway
      finetechGateway.SellStopLimit getProduct(), getQuantity(), getPrice(), getStopPx()
End Sub
```

The code above is from the sample FinetechExcelDemo.xlsm.



.NET Component

To use the .NET component, add CMEGatewayLib as a Reference in your C#, VB.NET or other project in a .NET supported language.

The gateway client is in the class CMEGatewayLib:: GatewayFIXClient. Note that all the other classes in this namespace are officially "un-documented". ***You should not rely on them continuing to exist or remaining unchanged.***

All the functions return bool and have an 'out string result' parameter. The return value: will be false on failure (result will be populated with reason), true on success.

The main difference between the .NET API and the COM and Win32 APIs, is that with the .NET API you are allowed to specify your own order numbers. The COM and Win32 APIs generate them for you.

Function Name	Description
Initialize	**public bool Initialize**(**out** string result, string connectionString) This function MUST be called first. See FTInitialize for explanation of connectionString.
GetVersion	**public** void **GetVersion**(**out** string version) Returns the underlying library version. Recommend this is displayed prominently to enable troubleshooting.
Start	**public bool Start**(**out** string result) Initiates the connection with the trade server.
SendNewOrder	**public bool SendNewOrder**(**out** string result, string fix) See FTSendNewOrder
SendNewOrderBuyMarket	**public bool SendNewOrderBuyMarket**(**out** string result, string id, string securityDesc, **int** qty) See FTSendNewOrderBuyMarket
SendNewOrderSellMarket	**public bool SendNewOrderSellMarket**(**out** string result, string id, string securityDesc, **int** qty) See FTSendNewOrderSellMarket

Function Name	Description
SendNewOrderBuyLimit	**public bool SendNewOrderBuyLimit**(**out** string result, string id, string securityDesc, **int** qty, string price) See FTSendNewOrderBuyLimit
SendNewOrderSellLimit	**public bool SendNewOrderSellLimit**(**out** string result, string id, string securityDesc, **int** qty, string price) See FTSendNewOrderSellLimit
SendNewOrderBuyStop	**public bool SendNewOrderBuyStop**(**out** string result, string id, string securityDesc, **int** qty, string price) See FTSendNewOrderBuyStop
SendNewOrderSellStop	**public bool SendNewOrderSellStop**(**out** string result, string id, string securityDesc, **int** qty, string price) See FTSendNewOrderSellStop
SendNewOrderBuyStopLimit	**public bool SendNewOrderBuyStopLimit**(**out** string result, string id, string securityDesc, **int** qty, string price, string stopPx) See FTSendNewOrderBuyStopLimit
SendNewOrderSellStopLimit	**public bool SendNewOrderSellStopLimit**(**out** string result, string id, string securityDesc, **int** qty, string price, string stopPx) See FTSendNewOrderSellStopLimit
CancelOrder	**public bool CancelOrder**(**out** string result, string id) See FTCancelOrder
CancelReplaceOrder	**public bool CancelReplaceOrder**(**out** string result, string id, string newId, **int** newQty, string newPrice, string newStopPx) See FTCancelReplaceOrder

The .NET component raises the following events:

Event Name	EventArgs type & Description

OnError	`public class ErrorEventArgs : System.EventArgs`
	`{`
	`public string Id;`
	`public string Text;`
	`}`
	
	Id – the order in error (if appropriate)
	Text – error text

OnNew	`public class NewEventArgs : System.EventArgs`
	`{`
	`public string Id;`
	`public string Symbol;`
	`public string SecurityDesc;`
	`public int OrderQty;`
	`}`
	
	An order is working,
	
	Id – order ID
	Symbol – instrument symbol, e.g. GC
	SecurityDesc – Security description, e.g. GCZ4
	OrderQty – quantity on the order

OnFill	`public class FillEventArgs : System.EventArgs`
	`{`
	`public string Id;`
	`public string Symbol;`
	`public string SecurityDesc;`
	`public int LastShares;`
	`public string LastPx;`
	`public int OrderQty;`
	`public int CumQty;`
	`public int LeavesQty;`
	`}`
	
	An order has been (partially) filled
	
	Id – order ID
	Symbol – instrument symbol, e.g. GC

Event Name	EventArgs type & Description
	SecurityDesc – Security description, e.g. GCZ4 OrderQty – quantity on the order LastShares – quantity this fill LastPx – price this fill CumQty – total quantity filled so far LeavesQty – quantity remaining open. If zero, order is no longer working

OnCanceled	public class CanceledEventArgs : System.EventArgs
	{
	public string Id;
	public string Symbol;
	public string SecurityDesc;
	}
	
	An order is no longer working
	
	Id – order ID
	Symbol – instrument symbol, e.g. GC
	SecurityDesc – Security description, e.g. GCZ4

OnConnection	The client is connected to the trade server
OnDisconnection	The connection to the trade server was lost.

END OF DOCUMENT.