OpenQuant FAQ (Frequently Asked Questions)

Table of Content

How to develop spread / pair trading strategies in OpenQuant

OpenQuant runs one instance of a strategy per instrument. This concept simplifies strategy development a lot since you always refer to one instrument (Strategy.Instrument) in your code and don't need to maintain arrays and tables holding different values (for example indicator) per instrument. It works pretty well for instrument independent staregies, i.e. for strategies that can run with one instrument, for example trend following or swing trading strategies, but you need to utilize certain techniques if you want to develop strategies that depend on two or more instruments, for example spread, pair trading. market neutral or correlation based strategies.

Assume we want to develop a strategy that buys qty of MSFT and sells qty of AAPL if MSFT / AAPL price ratio becomes larger that certain number X.

Code:

```
OnBar(Bar bar)
{
  Instrument instrument = Instruments["MSFT"];

  if (nstrument.Bar.Close / bar.Close > X)
  {
    Buy(qty);
    Sell(instrument, qty);
  }
}
```

You can also define instruments in the header of your strategy to speed things up

Code:

```
Instrument instrument1 = InstrumentManager.Instruments["MSFT"];
Instrument instrument2 = InstrumentManager.Instruments["AAPL"];

OnQuote(Quote quote)
{
  if (Instrument == instrument1)
    if (quote.Ask > instrument2.Quote.Bid * delta)
    {
       Buy(instrument1, qty);
       Sell(instrument2, qty);
     }
}
```

Note that you can also use OnBarSlice event handler.

Quote:

OnBarSlice fires whenever the market data provider BarFactory has finished emitting new bars for all the instruments in the current portfolio. BarFactory emits new bars sequentially when multiple instruments are being tradedâ€”first a bar for instrument 1, then a bar for instrument 2, and so on. After all individual bars have been emitted, BarFactory will fire OnBarSlice to tell strategy components that the end of the instrument list has been reached, and that all new bars for the current bar interval have been emitted. OnBarSlice solves the problem of strategy code not knowing when all the instrument bars for a correlation strategy are present and ready to be used. For example, if you try to do correlation calculations in the OnBar event handler for multiple instruments, your code will have to keep track of which instruments have been seen in the OnBar handler for the current bar time interval.

Code:

```
using System;
using System.Drawing;

using OpenQuant.API;
using OpenQuant.API.Indicators;

public class MyStrategy : Strategy
{
   Instrument DELL;
   Instrument CSCO;

   TimeSeries spread_series;

   [Parameter("Order quantity (number of contracts to trade)")]
   double Qty = 100;

   [Parameter("Length of SMA")]
   int SMALength = 20;

   [Parameter("Order of BBU")]
   double BBUOrder = 2;
```

```
[Parameter("Order of BBL")]
double BBLOrder = 2;

// indicators
BBU bbu;
BBL bbl;
SMA sma;

public override void OnStrategyStart()
{
   DELL = Instruments["DELL"];
   CSCO = Instruments["CSCO"];

   spread_series = new TimeSeries("DELL - CSCO", Color.Pink);

   // set up the moving averages
   sma = new SMA(spread_series, SMALength);
   sma.Color = Color.Yellow;
   Draw(sma, 2);
   // set up bollinger bands
   bbu = new BBU(spread_series, SMALength, BBUOrder);
   bbu.Color = Color.Green;
   bbl = new BBL(spread_series, SMALength, BBLOrder);
   bbl.Color = Color.Green;
   Draw(bbu, 2);
   Draw(bbl, 2);
   Draw(spread_series, 2);
}

public override void OnBarSlice(long size)
{
   if (Instrument == CSCO)
   {
      double spread = CSCO.Bar.Close / DELL.Bar.Close;
      spread_series.Add(CSCO.Bar.DateTime, spread);

      if (bbl.Count == 0)
         return;

      if (!HasPosition)
      {
         if (spread < bbl.Last)
         {
               Buy(CSCO, Qty);
               Sell(DELL, Qty);
            }
         }
         else
         {
            if (spread > bbu.Last)
            {
               Sell(CSCO, Qty);
               Buy(DELL, Qty);
            }
         }
      }
   }
}
```

How to use timer to cancel an order at a certain time

The code below demonstrates how to use strategy timer to cancel a limit order if the order is not filled within 20 seconds.

Note how Clock.Now is used to obtain current strategy time. This technique works in both simulation and live trading modes.

Code:

```
using OpenQuant.API;

public class MyStrategy : Strategy
{
   private Order order;
   private bool entry = true;

   public override void OnBar(Bar bar)
   {
      if (HasPosition)
         ClosePosition();

      if (entry)
      {
         order = LimitOrder(OrderSide.Buy, 100, bar.Close - 0.03);
```

```
            order.Send();

            AddTimer(Clock.Now.AddSeconds(20));

            entry = false;
         }
      }

      public override void OnTimer(DateTime signalTime)
      {
         if (!order.IsDone)
            order.Cancel();

         entry = true;
      }
}
```

How to pre-load historical data to a strategy

The code below pre-loads five last days of 1min bars from historical data base to strategy bar series on startup. Then it creates and draws SMA indicator on this series.

In this scenario you don't need to wait (128 minutes in this example) to run a strategy in live or paper trading mode. Indeed you should have historical data in the data base to use this technique. Note that you can capture real time data into historical data base when you run a strategy live, see http://www.smartquant.com/forums/viewto ... 7031#17031

Code:

```
using System;
using System.Drawing;

using OpenQuant.API;
using OpenQuant.API.Indicators;

public class MyStrategy : Strategy
{
   SMA sma;

   DateTime datetime1;
   DateTime datetime2;
   public override void OnStrategyStart()

   {

      datetime2 = DateTime.Now;

      datetime1 = datetime2.AddDays(-5);


      foreach (Bar bar in GetHistoricalBars(datetime1, datetime2, BarType.Time, 60))

         Bars.Add(bar);


      sma = new SMA(Bars, 128);


      Draw(sma);

   }

}
```

This code gets historical bars from the local OpenQuant data base. Note that you can modify the code so that it will download historical data from a historical data provider instead (for example IB).

Code:

```
GetHistoricalBars("IB", datetime1, datetime2, BarType.Time, 60);
```

How to use Scenario functionality (Select instruments)

This scenario shows how to filter and add the most liquid stocks into strategy project

Code:

```
public class MyScenario : Scenario

{

   public override void Run()

   {

      // get reference to strategy project


      Project project = Solution.Projects[0];


      // clear project instrument list


      project.ClearInstruments();


      // add most liquid stocks to instrument list


      foreach (Instrument instrument in InstrumentManager.Instruments)
```

```
         if (instrument.Type == InstrumentType.Stock)
         {
            BarSeries series = DataManager.GetHistoricalBars(instrument, BarType.Time,
86400);

            if (series.Count != 0 && series.Last.Volume > 50000000)
            {
               Console.WriteLine("Adding " + instrument);

               project.AddInstrument(instrument);
            }
         }

      // start backtest

      Start();
   }
}
```

How to use Scenario functionality (Optimization and Walk Forward analysis)

Here is a simple scenario that shows how to implement brute force optimization + walk-forward backtest on out-of-sample data interval. This scenario works with SMA crossover strategy.

Code:

```
public class MyScenario : Scenario
{
   public override void Run()
   {
      // set in-sample data interval

      Solution.StartDate = new DateTime(1995, 1, 1);
      Solution.StopDate  = new DateTime(2001, 1, 1);

      // get reference to strategy project

      Project project = Solution.Projects[0];

      // define variables
```

```
int best_length1 = 0;
int best_length2 = 0;
double best_objective = 0;

// brute force optimization loop

for (int length1 = 3;length1 <= 7;length1++)
   for (int length2 = 3;length2 <= 7;length2++)
      if (length2 > length1)
      {
         // set new parameters

         project.Parameters["Length1"].Value = length1;
         project.Parameters["Length2"].Value = length2;

         // print parameters

         Console.Write("Length1 = " + length1 + " Length2 = " + length2);

         // start backtest

         Start();

         // calculate objective function

         double objective = Solution.Portfolio.GetValue();

         // print objective

         Console.WriteLine(" Objective = " + objective);

         // check best objective

         if (objective > best_objective)
         {
            best_objective = objective;
            best_length1 = length1;
            best_length2 = length2;
         }
      }

// print best parameters
```

```
      Console.WriteLine("BEST PARAMETERS");
      Console.WriteLine();
      Console.WriteLine("SMA1 Length = " + best_length1);
      Console.WriteLine("SMA2 Length = " + best_length2);
      Console.WriteLine("Objective   = " + best_objective);

      // run strategy with the best parameters on out-of-sample data interval

      project.Parameters["Length1"].Value = best_length1;
      project.Parameters["Length2"].Value = best_length2;

      Solution.StartDate = new DateTime(2001, 1, 1);
      Solution.StopDate  = new DateTime(2005, 1, 1);

      Start();
   }
}
```

How to use Scenario functionality (Switch continuously from Simulation to Live mode)

The sample shows how to continuously switch from Simulation to Live mode using Scenario.

Code:

```
public class MyScenario : Scenario
{
   public override void Run()
   {
      // simulate historical data
      this.Solution.StartDate = new DateTime(1996, 1, 1);
      this.Solution.StopDate  = Clock.Now;

      Start(StrategyMode.Simulation);

      // make sure all objects: portfolios, orders,
      // indicators and strategy state remain the same
      // on next strategy start
      this.ResetOnStart = false;

      // start solution in Live mode
      Start(StrategyMode.Live);
   }
}
```

How to move common strategies functions to the base Strategy

If several strategies use the same functionality - it is good idea to move this functionality to the "base" Strategy and inherit other strategies from the "base" one. The example below shows how to create "Report" method, that simply outputs some text to Console, in the base strategy and use it later in derived strategies. It also shows how to implement "default" handlers like OnBar, OnQuote etc. The base MyStrategy1 class implements OnBar method, which can be overriden in derived strategies.

To create the base strategy - create a Class Library in Visual Studio .NET and add the following code to it:

Code:

```
using System;

using OpenQuant.API;

namespace MyStrategyLib
{
   public class MyStrategy1 : Strategy
   {
      public override void OnBar(Bar bar)
      {
         Report(Bar.ToString());
      }

      public void Report(string text)
      {
         Console.WriteLine(text);
      }
   }
}
```

Build the project and add a reference to this project output dll from OQ.

Then create a new project and paste the following code in the code.cs file:

Code:

```
using System;
using System.Drawing;

using OpenQuant.API;
using OpenQuant.API.Indicators;

using MyStrategyLib;

public class MyDerivedStrategy : MyStrategy1
{
   public override void OnBar(Bar bar)
   {
      Buy(100);

      base.OnBar(bar);
   }

   public override void OnPositionChanged()
   {
      Report("Position Qty " + Position.Qty);
   }
}
```

The strategy MyDerivedStrategy derives from base MyStrategy1 class. It overrides the OnBar method declated in MyStrategy1 class, so the MyStrategy1.OnBar method is called only because of the "base.OnBar(bar);" line. MyDerivedStrategy.OnPositionChanged handler code also calls Report method declared in MyStrategy1.

How to load and run solution from the command line

OpenQuant.exe [solution] [/r]

Example

OpenQuant.exe "C:\Users\Anton\Documents\OpenQuant\Solutions\Bollinger Bands\Bollinger Bands.oqs"

use /r command line parameter to run solution on startup

OpenQuant.exe "C:\Users\Anton\Documents\OpenQuant\Solutions\Bollinger Bands\Bollinger Bands.oqs" /r

How to setup a combo (multileg) instrument for IB

You need to set AltSymbol of the instrument using following format:

Symbol;Leg1;Leg2;...;LegN
where Leg is
ConId_Ratio_Action_Exchange_OpenClose_ShortSaleSlot_DesignatedLocation

for example
IBM;78491274_1_SELL_SMART_0_0_;81524101_1_BUY_SMART_0_0

How to add new instruments programmatically

The code below demonstrates how to write a simple script that adds new instruments to the instrument data base and application instrument list.

This example can be found in the Script Explorer among other script samples.

Code:

```
using System;

using OpenQuant.API;
using OpenQuant.API.Indicators;

public class MyScript : Script
{
   public override void Run()
   {
      string[] symbols = new string[] {"Stock1", "Stock2", "Stock3"};

      foreach (string symbol in symbols)
         new Instrument(InstrumentType.Stock, symbol);
   }
}
```

How to develop a custom market data or execution provider

You can find examples of custom market data and execution providers in OpenQuant installation folder (windows Start menu SmartQuant Ltd -> OpenQuant -> Samples -> Sample Providers). These examples include full source code for MBT, OEC and TT FIX providers that can be compiled under MSVS 2005. You can use this code as a pattern to develop interfaces to brokers and market data providers not supported out of the box.

Once you develop a new provider and compile it as dll in MSVS, you can add this new provider to OpenQuant via main menu -> Tools -> Options -> Configuration -> Providers -> User -> Add.

How to read/write to Excel file in the strategy code

The sample code below shows how to read from Excel file and write to it while a strategy is running. It reades two sma lengths from the SampleBook.xlsx file in the OnStrategyStart method and writes the last bar's close price to the Excel file in the OnBar method.

Code:

```
using System;
using System.Drawing;

using OpenQuant.API;
using OpenQuant.API.Indicators;

using Interop.Excel;

public class MyStrategy : Strategy
{
   SMA sma1;
   SMA sma2;

   Workbook workbook;
   Worksheet worksheet;

   public override void OnStrategyStart()
   {
      // open Excel file
      Application excel = new Application();

      workbook = excel.Workbooks.Open(@"C:\Users\s.b\Documents\SampleBook.xlsx", 0, false,
5, "", "", true, Interop.Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, null,
null);

      Sheets sheets = workbook.Worksheets;
      worksheet = (Worksheet)sheets.get_Item(1);

      // read A1 and A2 values
      double length1 = (double)worksheet.get_Range("A1", "A1").Value2;
      double length2 = (double)worksheet.get_Range("A2", "A2").Value2;
      //
      sma1 = new SMA(Bars, (int)length1);
      sma2 = new SMA(Bars, (int)length2);
   }

   public override void OnBar(Bar bar)
   {
      worksheet.get_Range("B1", "B1").Value2 = bar.Close;
   }

   public override void OnStrategyStop()
   {
      // save and close the workbook
      workbook.Save();
      workbook.Close(true, null, null);
   }
}
```

To build the sample you should add a reference to the Interop.Excel.dll, using the following steps:

- locate the Interop.Excel.dll file at "[Installation Directory]\Framework\bin" folder
- copy it to the "[Installation Directory]\bin" folder
- from the main menu go to Tools -> Options, in the opened Options dialog click "Projects and Solutions" node and select "Build" node. Click "Add" button, choose ".NET..." option and choose the Interop.Excel.dll file.

How to develop a custom indicator

OpenQuant offers a possibility to develop custom user indicators and use such user indicators in OpenQuant strategy.

You can develop a custom indicator as a dll in MS Visual Studio and add to OpenQuant as a reference. You can exchange your custom indicators with another OpenQuant users this way (without need to disclose indicator code if you don't want to). Alternatively, you can add indicator code right before strategy code in OpenQuant editor and use it in this strategy.

In order to start writing a user indicator you should add

Code:

```
using OpenQuant.API.Plugins;
```

Any user indicator should derive from the UserIndicator class defined in the OpenQuant.API.Plugins namespace.

Developing a user indicator in the OpenQuant framework is quite simple. All you need to do is to override the Calculate(int index) method of the UserIndicator class using Input property of UserIndicator class.

The Input property is a reference to indicator input time series. It has ISeries type, so that it can be BarSeries, TimeSeries or another Indicator (this way you can develop indicator of indicator).

The Calculate(int index) method should return indicator value calculated for the index-th element in the Input series. It should return Double.NaN if there is no indicator value for the index-th element of the Input series (usually in case of indicators that require several past values of Input time series, for example simple moving average or momentum indicators).

Let's develop a simple Accumulated Money Flow (AMF) indicator as an example. The simplest formula will be making an upday volume negative and a down day volume positive and then adding up all the buy/sell volume from date from which the data became available.

We create a new class that we call AMF and derive this class from UserIndicator class. Then we add a constructor of AMF class that takes BarSeries as input.

Code:

```
public class AMF: UserIndicator
{
   public AMF(BarSeries input) : base(input)
   {
      Name = "AMF";
   }
}
```

Note that we use the Name property of te UserIndicator class to give a name to our indicator. This name will be displayed on the Bar chart and elsewhere.

Now if we write

Code:

```
AMF indicator = new AMF(Bars);
```

in our strategy, we will create an AMF indicator called "AMF" and Input property assigned to the Bars series of our strategy.

Now we need to override the Calculate method of our AMF indicator.

Code:
```
public class AMF: UserIndicator
{
   double volume = 0;

   public AMF(BarSeries input) : base(input)
   {
      Name = "AMF";
   }

   public override double Calculate(int index)
   {
      if (index > 1)
      {
         if (Input[index, BarData.Close] > Input[index-1, BarData.Close])
            volume += Input[index, BarData.Volume];
         else
            volume -= Input[index, BarData.Volume];

         return volume;
      }
      else
         return Double.NaN;
   }
}
```

Note that we can use two techniques to work with the Input property. We can either cast it to a specific series type and use this type in our calculations, for example

Code:
```
BarSeries bars = Input as BarSeries;

double close = bars[index].Close;
```

or we can use common indexers of ISeries interface.

Code:

```
double close = Input[index, BarData.Close];
```

We use the latter techniques in our Calculation method.

The rest of Calculate code is quite simple. First we check that we have enough elements in the Input series to access previous element (bar) of the Input series. We return Double.NaN if there is no previous bar. If we do have a previous bar, we compare closing prices of the current (index) bar and previous (index - 1) bar and then adjust volume class variable accordingly.

Now we can add our indicator to a strategy and draw it on the strategy bar chart.

Code:

```
using System;
using System.Drawing;

using OpenQuant.API;
using OpenQuant.API.Indicators;
using OpenQuant.API.Plugins;

public class AMF: UserIndicator
{
   double volume = 0;

   public AMF(BarSeries input) : base(input)
   {
      Name = "AMF";
   }

   public override double Calculate(int index)
   {
      if (index > 1)
      {
         if (Input[index, BarData.Close] > Input[index-1, BarData.Close])
            volume += Input[index, BarData.Volume];
         else
            volume -= Input[index, BarData.Volume];

         return volume;
      }
      else
         return Double.NaN;
   }
}

public class MyStrategy : Strategy
{
   public override void OnStrategyStart()
   {
      AMF AMF = new AMF(Bars);

      AMF.Color = Color.White;

      Draw(AMF, 2);
   }

   public override void OnBar(Bar bar)
   {
   }
}
```

Note that there are several other examples of user indicators included in the OpenQuant installation. They can be found in the Samples folder in OpenQuant Windows menu.

How to submit a bracket order

The code below uses OnOrderFilled event handler to submit legs of bracket orders and OCA (One Cancels All) group to cancel other leg of bracket order if take profit or stop loss order gets executed.

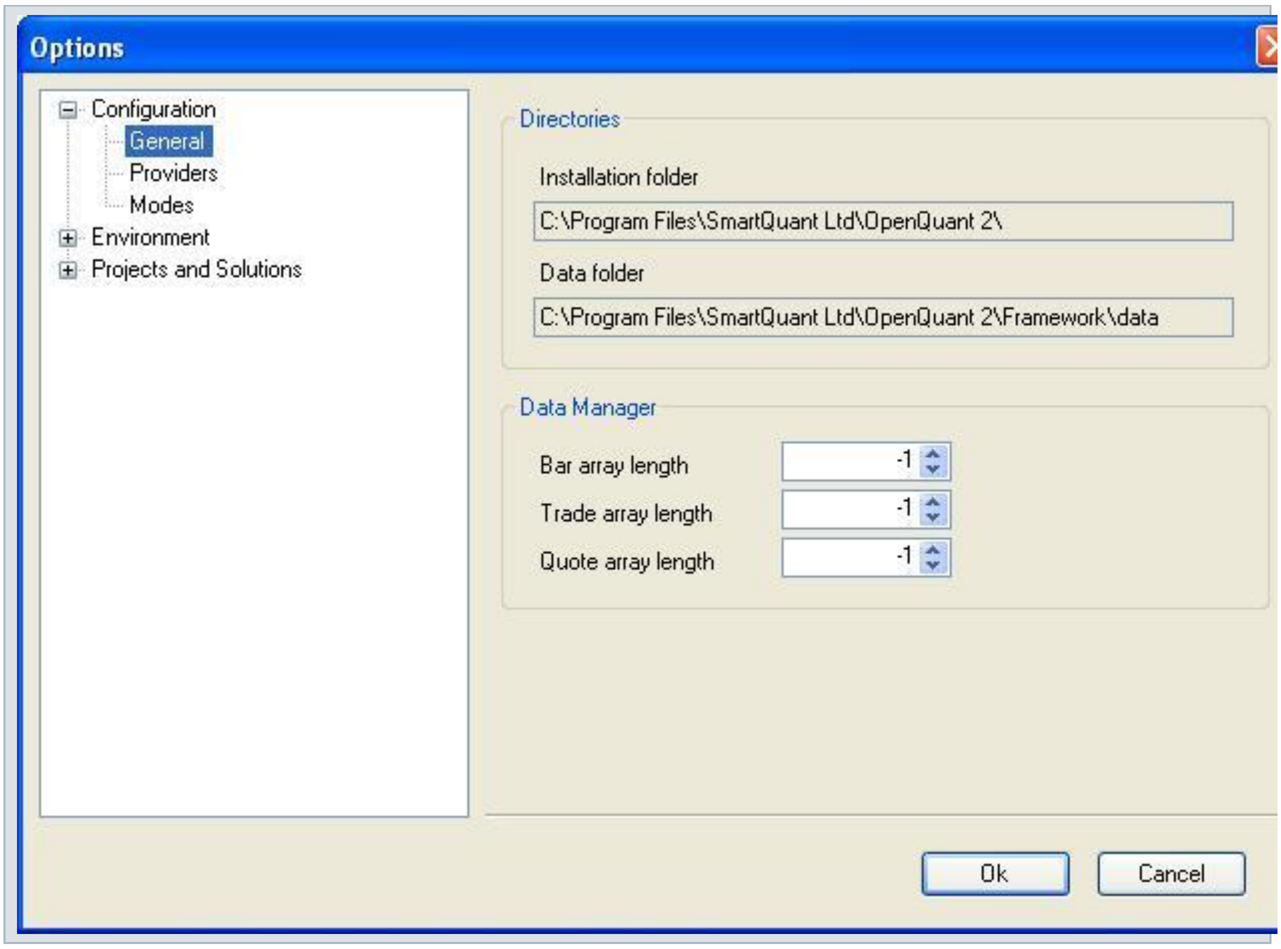
Code:

```
Order buyOrder;
Order sellOrder1;
Order sellOrder2;

int OCACount = 0;

public override void OnBar(Bar bar)
{
  if (EntryCondition)
  {
    buyOrder = BuyLimitOrder(qty, 100);
    buyOrder.Send();
  }
}

public override void OnOrderFilled(Order order)
{
  if (order == buyOrder)
  {
    OCACount++;

    sellOrder1 = SellLimitOrder(qty, 110);
    sellOrder2 = SellStopOrder(qty, 90);

    sellOrder1.OCA = "OCA " + OCACount;
    sellOrder2.OCA = "OCA " + OCACount;

    sellOrder1.Send();
    sellOrder2.Send();
  }
}
```

How to process large amounts of historical data

When you backtest a strategy, all incoming market data is stored in memory in Strategy.Bars, Strategy.Quotes and Strategy.Trades arrays. This may cause an out of memory exception if you process large enough amounts of historical data. You can set fixed sizes of these in-memory arrays, so that only last N bars, quotes or trades will be stored.

You can navigate to Tools->Options in the OpenQuant 2.x main menu and set desired number of stored market data entries. Use "-1" value to store all incoming market data.


Options

Directories

Installation folder: C:\Program Files\SmartQuant Ltd\OpenQuant 2\

Data folder: C:\Program Files\SmartQuant Ltd\OpenQuant 2\Framework\data

Data Manager

Bar array length: -1

Trade array length: -1

Quote array length: -1

How to capture live market data to historical data base

The code below shows how to capture market (bar) data while running a strategy in live or paper trading mode.

Code:

```
using OpenQuant.API;

public class MyStrategy : Strategy
{
   public override void OnBar(Bar bar)
   {
      DataManager.Add(Instrument, bar);
   }
}
```

Note that you can capture quote and trade data as well by changing or adding corresponding methods

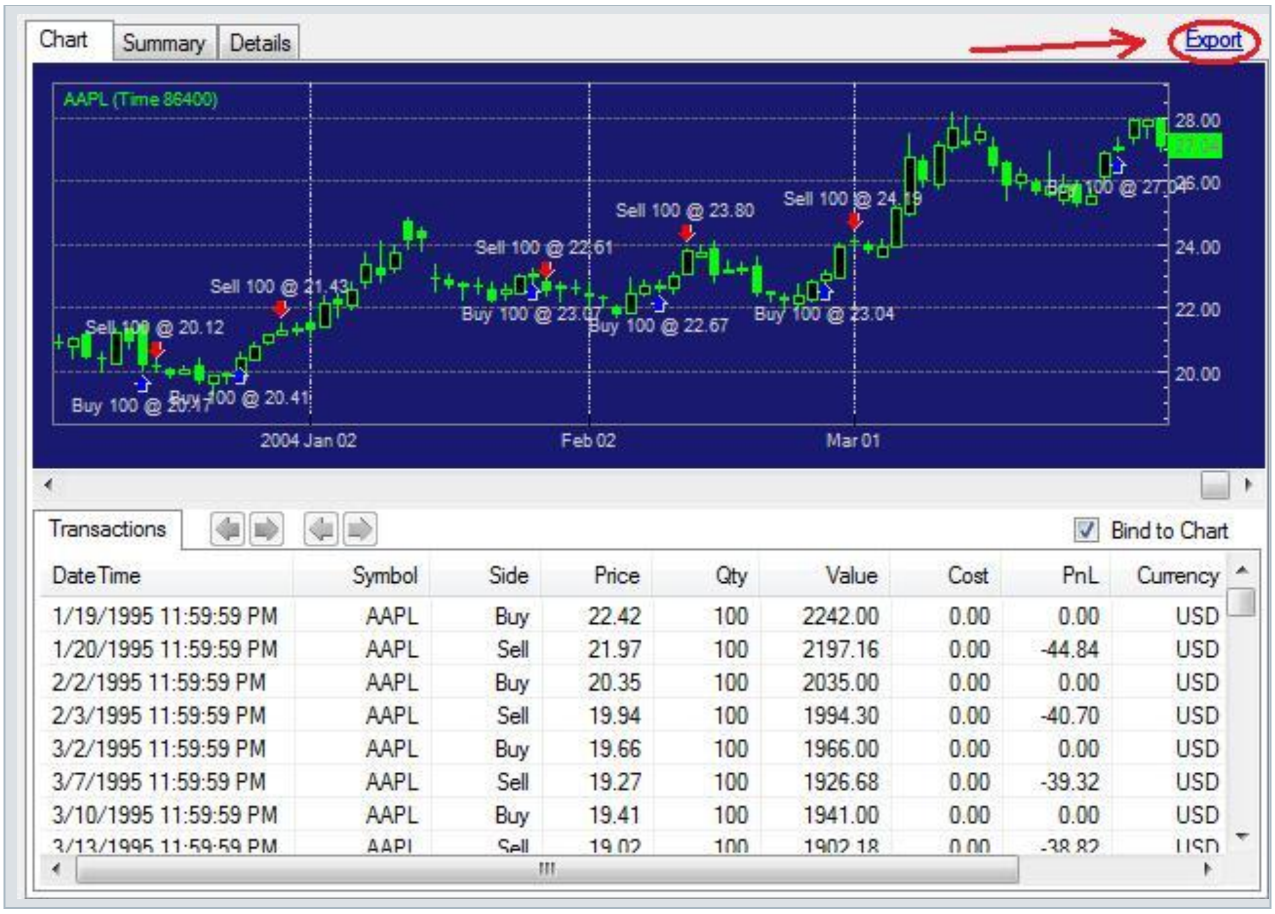
Code:

```
using OpenQuant.API;

public class MyStrategy : Strategy
{
   public override void OnQuote(Quote quote)
   {
      DataManager.Add(Instrument, quote);
   }
}
```

How to Export Postitions, Transactions and Statistics to CSV

To export portfolio positions, transactions and statistcs to CSV file you should open the Results window and click the blue link "Export" in the right top corner of the window.

To open the Results window: right click the Solution or Project node in the Solution Explorer window and click "View Results".



DateTime	Symbol	Side	Price	Qty	Value	Cost	PnL	Currency
1/19/1995 11:59:59 PM	AAPL	Buy	22.42	100	2242.00	0.00	0.00	USD
1/20/1995 11:59:59 PM	AAPL	Sell	21.97	100	2197.16	0.00	-44.84	USD
2/2/1995 11:59:59 PM	AAPL	Buy	20.35	100	2035.00	0.00	0.00	USD
2/3/1995 11:59:59 PM	AAPL	Sell	19.94	100	1994.30	0.00	-40.70	USD
3/2/1995 11:59:59 PM	AAPL	Buy	19.66	100	1966.00	0.00	0.00	USD
3/7/1995 11:59:59 PM	AAPL	Sell	19.27	100	1926.68	0.00	-39.32	USD
3/10/1995 11:59:59 PM	AAPL	Buy	19.41	100	1941.00	0.00	0.00	USD
3/13/1995 11:59:59 PM	AAPL	Sell	19.02	100	1902.18	0.00	-38.82	USD

How to import several CSV data files in a script

Code:

```
using System;
using System.IO;

using OpenQuant.API;

public class MyScript : Script
{
   public override void Run()
   {
      const string dataDir = @"D:\Data\CSV\";

      long dailyBarSize = 86400;

      string[] filenames = new string[]
         {
            "AAPL.csv",
            "CSCO.csv",
            "MSFT.csv"
         };

      // CSV data files have invariant date/number format
      System.Globalization.CultureInfo culture =
System.Globalization.CultureInfo.InvariantCulture;

      foreach (string filename in filenames)
      {
         Console.WriteLine();

         string path = dataDir + filename;

         // check file exists
         if (!File.Exists(path))
         {
            Console.WriteLine(string.Format("File {0} does not exists.", path));

            continue;
         }

         Console.WriteLine(string.Format("Processing file {0} ...", path));

         // get instrument
         string symbol = filename.Substring(0, filename.IndexOf('.'));

         Instrument instrument = InstrumentManager.Instruments[symbol];

         if (instrument == null)
         {
            Console.WriteLine(string.Format("Instrument {0} does not exist.", symbol));

            continue;
         }

         // read file and parse data
         StreamReader reader = new StreamReader(path);

         reader.ReadLine(); // skip CSV header

         string line = null;
         while ((line = reader.ReadLine()) != null)
         {
            string[] items = line.Split(',');

            // parse data
            DateTime date = DateTime.ParseExact(items[0], "yyyy-M-d", culture);

            double high  = double.Parse(items[1], culture);
            double low   = double.Parse(items[2], culture);
            double open  = double.Parse(items[3], culture);
            double close = double.Parse(items[4], culture);

            long volume  = long.Parse(items[5], culture);

            // add daily bar
            DataManager.Add(
               instrument,
               date,
               open,
               high,
               low,
               close,
               volume,
               dailyBarSize);
         }

         reader.Close();

         //
         Console.WriteLine(string.Format("CSV data for {0} was successfully imported.",
instrument.Symbol));
      }
   }
}
```

Here is the version of this code that I use that grabs the Directory info & Filenames. It works pretty well so far.

Code:

```
using System;
using System.IO;

using OpenQuant.API;

public class MyScript : Script
{
   public override void Run()
   {
      const string dataDir = @"C:\ua\Files\OQ_Import\";

      long dailyBarSize = 86400;

      //Lookup Directory & File Info
      DirectoryInfo directory_Info = new DirectoryInfo(@"C:\ua\Files\OQ_Import\");
      FileInfo[] file_Info = directory_Info.GetFiles("*.csv");

      // CSV data files have invariant date/number format
      System.Globalization.CultureInfo culture =
System.Globalization.CultureInfo.InvariantCulture;

      foreach (FileInfo file_info in file_Info)
      {
         Console.WriteLine(file_info.Name.Substring(0, file_info.Name.IndexOf('.')));

         Console.WriteLine();

         string filename = file_info.Name.Substring(0, file_info.Name.IndexOf('.'));
         string path = dataDir + filename + ".csv";

         // check if file exists
         if (!File.Exists(path))
         {
            Console.WriteLine(string.Format("File {0} does not exists.", path));

            continue;
         }

         Console.WriteLine(string.Format("Processing file {0} ...", path));

         // get instrument
         string symbol = filename; //.Substring(0, filename.IndexOf('.'));
```

```
Instrument instrument = InstrumentManager.Instruments[symbol];

if (instrument == null)
{
   Console.WriteLine(string.Format("Instrument {0} does not exist.", symbol));

   continue;
}

// read file and parse data
StreamReader reader = new StreamReader(path);

reader.ReadLine(); // skip CSV header

string line = null;

while ((line = reader.ReadLine()) != null)
{
   string[] items = line.Split(',');

   // parse data
   DateTime date = DateTime.ParseExact(items[0], "yyyyMMdd", culture);

   double open  = double.Parse(items[3], culture);
   double high  = double.Parse(items[4], culture);
   double low   = double.Parse(items[5], culture);
   double close = double.Parse(items[6], culture);
   long volume  = long.Parse(items[7], culture);

   // add daily bar
   DataManager.Add(
      instrument,
      date,
      open,
      high,
      low,
      close,
      volume,
      dailyBarSize);
}

reader.Close();
```

```
         //
         Console.WriteLine(string.Format("CSV data for {0} was successfully imported.",
instrument.Symbol));
      }
   }
}
```

How to Setup FOREX Instruments for IB

If you want to set up two FOREX instruments in OpenQuant, for example EUR/USD and EUR/GBP currency pairs, IB TWS would require to set both instrument names to EUR and set instrument currency to USD for the first instrument and to GBP for the second one.

Since Instrument.Symbol serves as unique instrument identifier in OpenQuant, it's not possible to have two instruments with Symbol = EUR. Thus we should use AltSymbol technique.

For the EUR/USD pair we create an instrument with Symbol = EUR USD and set AltSource = IB and AltSymbol = EUR. We set Currency = USD and Exchange = IDEALPRO.

For the EUR/GBP pair Symbol = EUR GBP, AltSource = IB, AltSymbol = EUR, Currency = USD and Exchange = IDEALPRO.

How to Debug OpenQuant Strategies in Visual Studio

This demo video demonstrates how to debug OpenQuant strategies in Microsoft Visual Studio.

http://www.smartquant.com/openquant/vid ... /msvs.html

How to use different time frames in a strategy

The code below shows how to access series of 1 and 5 minute bars in a strategy. It's assumed that you have 1 and 5 min bars added to the market data folder of your strategy.

The code also shows how to filter bars coming into OnBar event handler using bar.Size property.

Code:

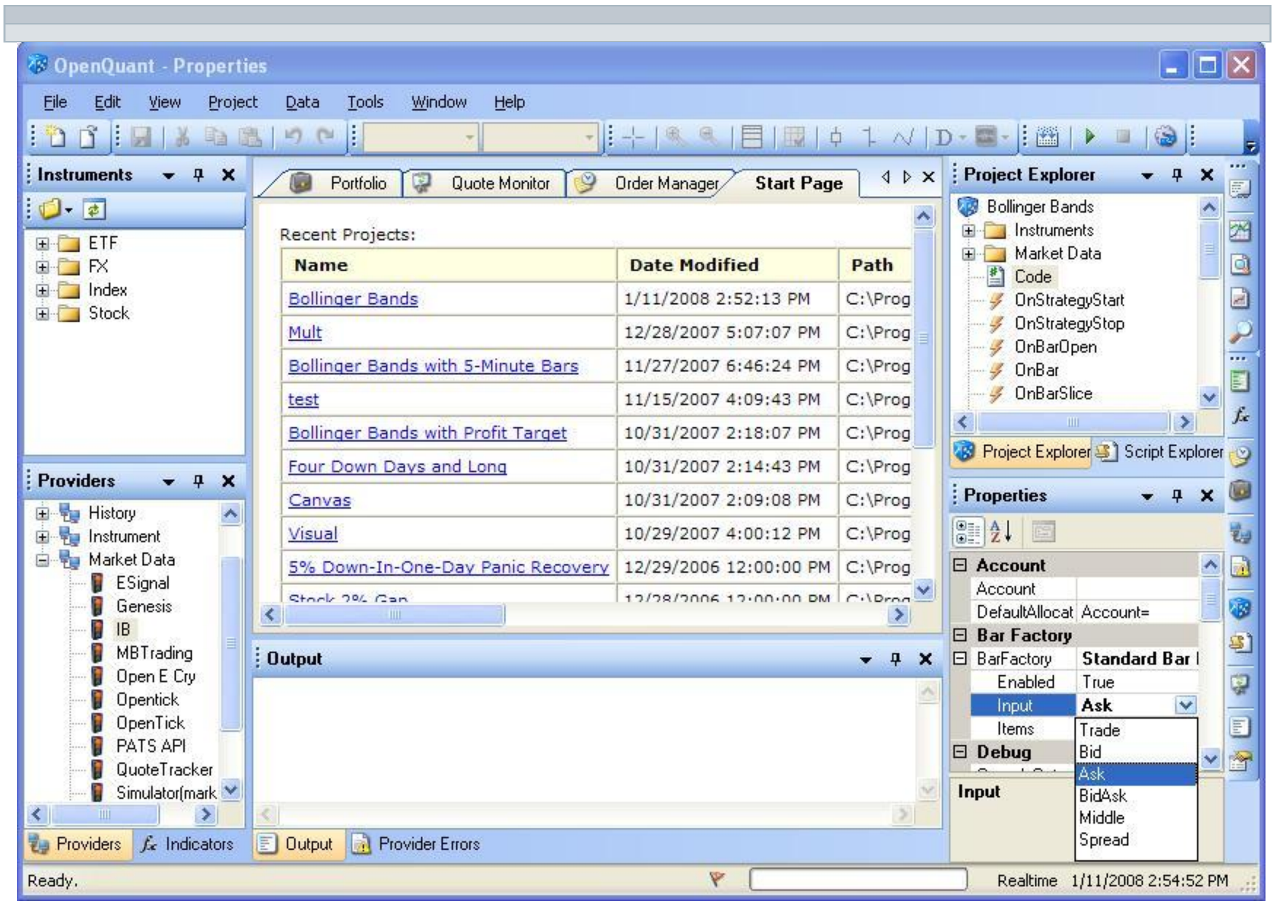
```
   public override void OnBar(Bar bar)
   {
```

```
   BarSeries bars1min = GetBars(60);

   BarSeries bars5min = GetBars(300);


   if (bar.Size == 300)

   {

      if (bar.Close > bars1min.Ago(2).Close)

          Buy(100);

   }

}
```

How to build bars from quotes (IB / FOREX)

Trade data is not always available from a data provider (FOREX on IB is a typical case). In this situation the only possibility to run a strategy requiring bar data is to build bars from quotes.

- click on IB market data provider in the Providers window
- switch to Properties window
- expand BarFactory section
- select alternative Input (bid, ask, bidask, middle, spread) for the BarFactory



Recent Projects:

Name	Date Modified	Path
Bollinger Bands	1/11/2008 2:52:13 PM	C:\Prog
Mult	12/28/2007 5:07:07 PM	C:\Prog
Bollinger Bands with 5-Minute Bars	11/27/2007 6:46:24 PM	C:\Prog
test	11/15/2007 4:09:43 PM	C:\Prog
Bollinger Bands with Profit Target	10/31/2007 2:18:07 PM	C:\Prog
Four Down Days and Long	10/31/2007 2:14:43 PM	C:\Prog
Canvas	10/31/2007 2:09:08 PM	C:\Prog
Visual	10/29/2007 4:00:12 PM	C:\Prog
5% Down-In-One-Day Panic Recovery	12/29/2006 12:00:00 PM	C:\Prog
Stock 2% Gap	12/28/2006 12:00:00 PM	C:\Prog

Account	
Account	
DefaultAllocat	Account=
Bar Factory	
BarFactory	Standard Bar I
Enabled	True
Input	Ask
Items	
Debug	

Input: Trade, Bid, Ask, BidAsk, Middle, Spread

Input

How to access position of a different instrument

A Strategy has a reference to Portfolio object, which you can use to access Position of any instrument using Portfolio.Positions list of portfolio position.

Code:
```
if (Portfolio.Positions[instrument] != null)
{
  double qty = Portfolio.Positions[Instrument].Qty;

  ...
}
```

You can also use Portfolio.HasPosition(instrument) to simplify this code.

How a strategy can understrand if it is running live?

You can use Strategy.Mode property to understand if your strategy code is running in live, paper trading or simulation mode. This way you can turn off certain functionality (for example heavy debugging) if you are running in the simulation mode.

Code:
```
public override void OnBar(Bar bar)
{
    if (Mode != StrategyMode.Simulation)
      Console.WriteLine("Close = " + bar.Close);
}
```

How to run OpenQuant under Windows Vista

Windows Vista problem:

"This software requires administrative privileges to run."

There are 2 ways to resolve this problem:

1. Disable UAC (User Account Control) mechanism. (not recommended,
because your computer will have not enough protection from viruses,
etc.

2. Find OpenQuant.exe in the installation folder.

(usually C:\Program Files\SmartQuant Ltd\OpenQuant)

a) Right-click on the program and select 'Run as Administrator'.

or

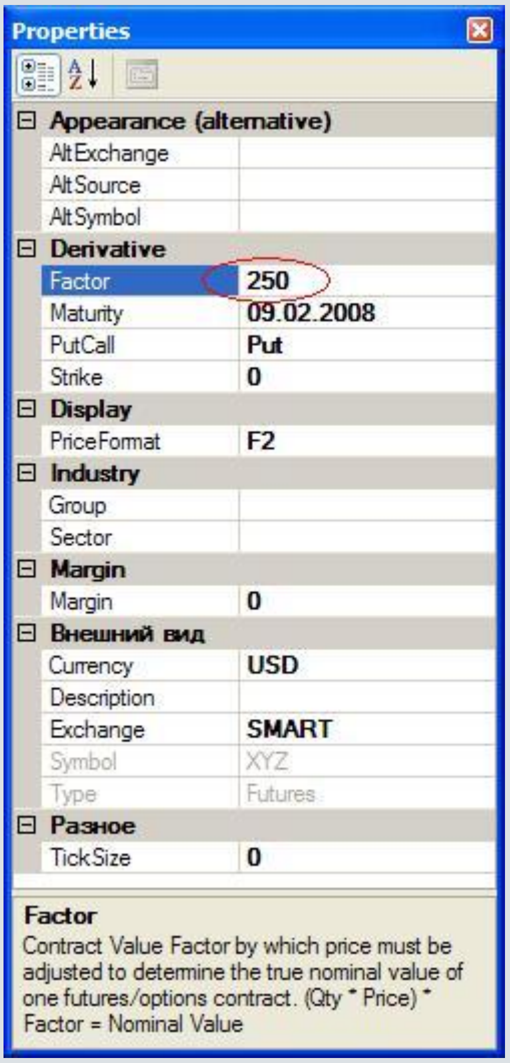
b) Right-click on the program, select Send To-> Desktop (create shortcut). Then right-click on the shortcut, select Properties and check 'Run as Administrator' box in Advanced tab. Double-click shortcut to run OpenQuant.

How to set BigPoint value for futures and currency contracts

You can use Instrument.Factor for this.

Position.Value = Instrument.Price * Instrument.Factor * Position.Qty

"Factor" terminology comes from FIX (Financial Information eXchange) protocol, www.fixprotocol.org, which we follow in OpenQuant business object model.


Properties

Appearance (alternative)	
AltExchange	
AltSource	
AltSymbol	
Derivative	
Factor	250
Maturity	09.02.2008
PutCall	Put
Strike	0
Display	
PriceFormat	F2
Industry	
Group	
Sector	
Margin	
Margin	0
Внешний вид	
Currency	USD
Description	
Exchange	SMART
Symbol	XYZ
Type	Futures
Разное	
TickSize	0

Factor
Contract Value Factor by which price must be adjusted to determine the true nominal value of one futures/options contract. (Qty * Price) * Factor = Nominal Value

How to send orders to different IB FA accounts

Code:

```
public override void OnBar(Bar bar)
{
Order order1 = LimitOrder(OrderSide.Buy, 1, bar.Close - 1);
Order order2 = LimitOrder(OrderSide.Sell, 1, bar.Close + 1);

order1.Account = "DU16151";
order2.Account = "DU16152";

order1.Send();
order2.Send();
}
```