ComponentOne

FinancialChart for WPF

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

Trademarks

Warranty

Copying and Distribution

Table of Contents

FinancialChart for WPF Overview

FinancialChart for WPF is an easy-to-use visualization control that enables you to create stock trending charts. The control lets you perform technical analysis in financial applications using predefined financial indicators and special chart types. In addition, the control allows you to analyze and interact with data trends using analytics, such as trendlines and moving average and range selector.

Below is a full listing of the sections that provide an extensive coverage on FinancialChart for WPF:

- Key Features
- Quick Start
- Financial Chart Types
- Analytics
- Interaction

Getting Started with WPF Edition

For information on installing **ComponentOne Studio WPF Edition**, licensing, technical support, namespaces and creating a project with the control, visit Getting Started with WPF Edition.

Key Features

- **Chart Type**: Change a line chart to a scatter chart or any other chart type by setting the ChartType property. Provides fifteen different chart types to choose from.
- **Range Selector**: Adjust the FinancialChart's visible range of data at runtime.
- **Trend Lines**: Visualize trends in data and analyze the problems of prediction.
- **Tooltips**: Display chart values using tooltips.
- **Header and Footer**: Use simple properties to set a title and footer text.
- **Legend**: Change position of the legend as needed.
- **Line Break**: Illustrate the price changes of an asset or market using lines or vertical boxes.
- **Annotations**: Mark important events or news attached to a specific data point on financial charts
- **Moving Average**: Analyze data points through a series of averages of different subsets of entire data set.
- **Indicators**: Analyze and predict trends in price and volume momentum of trading instruments using technical indicators, such as Average True Range, Relative Strength Index, Commodity Channel Index, and Williams %R.

Quick Start

This quick start illustrates the process of creating a simple application using **FinancialChart for WPF** and running the same in Visual Studio.

Perform the following steps to walk through the FinancialChart control quickly:

1. Adding FinancialChart to the Form
2. Binding FinancialChart to a Data Source
3. Running the Application

Step 1: Adding FinancialChart to the Application

This step creates a new Visual Studio project and adds the FinancialChart control to it.

1. Create a new **WPF Application** in Visual Studio.
    1. Select **File | New | Project**. The **New Project** dialog box appears.
    2. In the **New Project** dialog box, select a language in the left-hand pane, and then select **WPF Application** from the list of applications in the center pane.
    3. Give your application a **Name**, and then select **OK**.
2. Open the **MainWindow.xaml** file.
3. Place your cursor between the **<Grid> </Grid>** tags within either your Window or your UserControl, depending on the type of application you've created.
4. Locate the **C1FinancialChart** control in Visual Studio's **ToolBox**. Double-click the control to add it to your application. The following references are added to the project:
   C1.WPF.4.dll
   C1.WPF.FinancialChart.4.dll
   C1.WPF.FlexChart.4.dll
   If the references are not added, you need to add the same manually - right-click the **References** folder in the **Solution Explorer** and select **Add | New Reference**.

   The XAML markup resembles the following:
    - **XAML**

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="FinancialChart.MainWindow"
        Title="MainWindow" Height="387.285" Width="641.667">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             ChartType="HeikinAshi"
                             HorizontalAlignment="Left"
                             Height="325" VerticalAlignment="Top"
                             Width="523">
            <c1:FinancialSeries AxisX="{x:Null}" AxisY="{x:Null}"
                                Binding="High,Low,Open,Close,Volume"
                                BindingX="Date"
                                Chart="{x:Null}"
                                SeriesName="{x:Null}">
                <c1:FinancialSeries.ItemsSource>
                    <c1:QuoteCollection>
                        <c1:Quote Close="23.23" Date="01/23/15"
                                  High="24.73" Low="20.16"
                                  Open="20.2" Volume="42593223"/>
```

```
                                    <c1:Quote Close="22.6" Date="01/26/15"
                                              High="24.39" Low="22.5"
                                              Open="23.67" Volume="8677164"/>
                                    <c1:Quote Close="21.3" Date="01/27/15"
                                              High="22.47" Low="21.17"
                                              Open="22" Volume="3272512"/>
                                    <c1:Quote Close="19.78" Date="01/28/15"
                                              High="21.84" Low="19.6"
                                              Open="21.62" Volume="5047364"/>
                                    <c1:Quote Close="18.8" Date="01/29/15"
                                              High="19.95" Low="18.51"
                                              Open="19.9" Volume="3419482"/>
                                </c1:QuoteCollection>
                            </c1:FinancialSeries.ItemsSource>
                        </c1:FinancialSeries>
                    </c1:C1FinancialChart>

                </Grid>
            </Window>
```

The FinancialChart control is successfully added to the application.

Step 2: Binding FinancialChart to a Data Source

This step binds the FinancialChart control to a valid data source.

1. Create the data source as follows:
    1. Right-click the project and select **Add | Class.**
    2. Select **Class** from the list of templates, name it as **DataService.cs**, and click **Add**.
    3. Add the following code in **DataService** class to generate the data.
    - **Visual Basic**

```
Public Class DataService

    Public Shared Function CreateData() As List(Of DataItem)
        Dim data = New List(Of DataItem)()

        Dim dt As DateTime = DateTime.Today

        data.Add(New DataItem(dt.[Date], 79))
        data.Add(New DataItem(dt.[Date].AddDays(-7), 78))
        data.Add(New DataItem(dt.[Date].AddDays(-14), 73))
        data.Add(New DataItem(dt.[Date].AddDays(-21), 74))
        data.Add(New DataItem(dt.[Date].AddDays(-28), 76))
        data.Add(New DataItem(dt.[Date].AddDays(-35), 74))
        data.Add(New DataItem(dt.[Date].AddDays(-42), 75))
        data.Add(New DataItem(dt.[Date].AddDays(-49), 75))
        data.Add(New DataItem(dt.[Date].AddDays(-56), 80))
        Return data
    End Function

End Class

Public Class DataItem
    Public Sub New(date__1 As DateTime, sales__2 As Integer)
        [Date] = date__1
        Sales = sales__2
    End Sub

    Public Property [Date]() As DateTime
```

```
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As DateTime
    Public Property Sales() As Integer
        Get
            Return m_Sales
        End Get
        Set
            m_Sales = Value
        End Set
    End Property
    Private m_Sales As Integer
End Class
```

- **C#**

```
class DataService
{
    public static List<DataItem> CreateData()
    {
        var data = new List<DataItem>();

        DateTime dt = DateTime.Today;

        data.Add(new DataItem(dt.Date,79));
        data.Add(new DataItem(dt.Date.AddDays(-7), 78));
        data.Add(new DataItem(dt.Date.AddDays(-14), 73));
        data.Add(new DataItem(dt.Date.AddDays(-21), 74));
        data.Add(new DataItem(dt.Date.AddDays(-28), 76));
        data.Add(new DataItem(dt.Date.AddDays(-35), 74));
        data.Add(new DataItem(dt.Date.AddDays(-42), 75));
        data.Add(new DataItem(dt.Date.AddDays(-49), 75));
        data.Add(new DataItem(dt.Date.AddDays(-56), 80));
        return data;
    }
}

public class DataItem
{
    public DataItem(DateTime date, int sales)
    {
        Date = date;
        Sales = sales;
    }

    public DateTime Date { get; set; }
    public int Sales { get; set; }
}
```

2. Bind the data to FinancialChart as follows:

    1. Edit the **<Grid>** tag to the following markup to provide data to FlexChart:
        - **MainWindow.xaml**

```
<Grid>
    <Finance:C1FinancialChart x:Name="financialChart"
                              ChartType="LineSymbols"
                              ItemsSource="{Binding DataContext.Data}"
                              HorizontalAlignment="Left"
```

```
                                      Height="321"
                                      VerticalAlignment="Top"
                                      Width="620"
                                      Margin="81,94,0,0">
            <Finance:FinancialSeries AxisX="{x:Null}"
                                         AxisY="{x:Null}"
                                         Binding="Sales"
                                         BindingX="Date"
                                         Chart="{x:Null}"
                                         SeriesName="{x:Null}">
            </Finance:FinancialSeries>
        </Finance:C1FinancialChart>
</Grid>
```

To specify the binding source, you need to add the **DataContext = "{Binding RelativeSource={RelativeSource Mode=Self}}"** markup in the **<Window>** tag of the **MainWindow.xaml** file.

2. Switch to **Code View**. Add the following code in the MainWindow class to plot the data in the chart
   - **MainWindow.xaml.vb**

```
Partial Public Class MainWindow
    Inherits Window
    Private _data As List(Of DataItem)

    Public Sub New()
        Me.InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of DataItem)
        Get
            If _data Is Nothing Then
                _data = DataService.CreateData()
            End If

            Return _data
        End Get
    End Property
End Class
```

   - **MainWindow.xaml.cs**

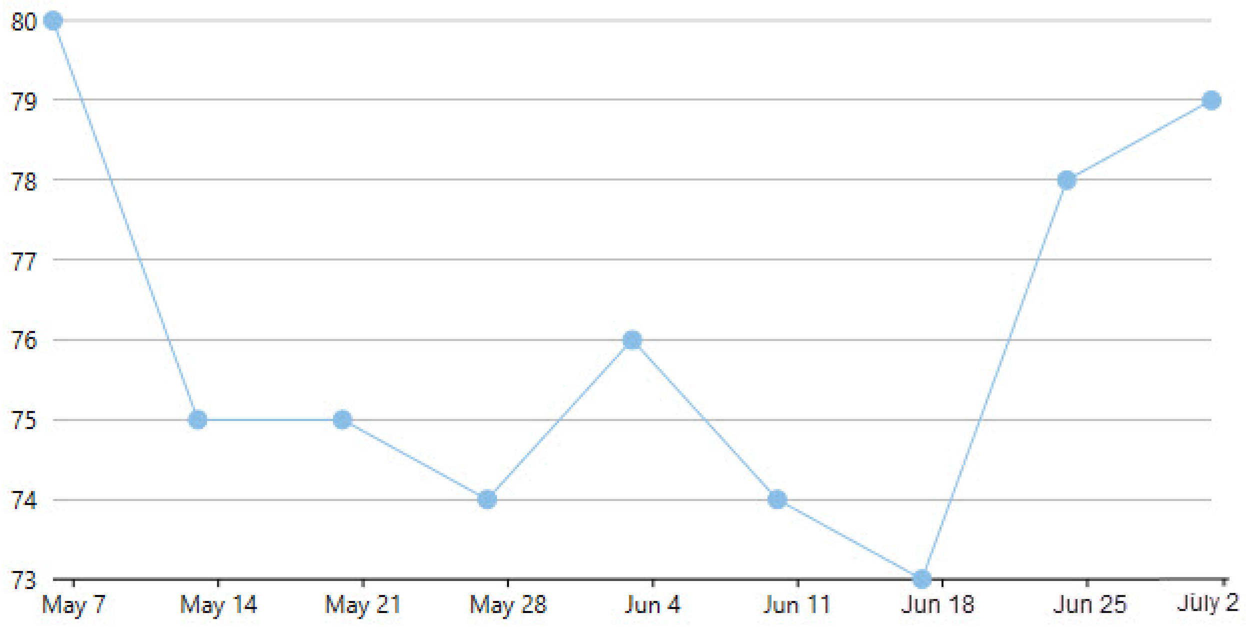
```
public partial class MainWindow : Window
{
    private List<DataItem> _data;

    public MainWindow()
    {
        this.InitializeComponent();
    }

    public List<DataItem> Data
    {
        get
        {
            if (_data == null)
            {
                _data = DataService.CreateData();
            }

            return _data;
        }
    }
}
```

The FinancialChart control is successfully bound to the data source.

Step 3: Running the Application

This step runs the project and observes the output.

The following output appears once the project is run.



You have successfully created a simple FinancialChart application. This concludes quick start.

Financial Chart Types

FinancialChart provides you with 15 chart types to cater to your each and every financial data visualization requirement. You can set the chart type of the FinancialChart control by setting the ChartType property, which accepts values from the FinancialChartType enumeration.

The following table lists all chart types provided by FinancialChart:

Area chart	**ArmsCandleVolume chart**	**Candlestick chart**	**CandleVolume chart**
Column chart	**ColumnVolume chart**	**EquiVolume chart**	**HeikinAshi chart**
HighLowOpenClose chart	**Kagi chart**	**Line chart**	**Line Break chart**
Line Symbols chart	**Renko chart**	**Scatter chart**	**Point and Figure chart**

The following code snippet sets the ChartType property.

XAML

```
<c1:C1FinancialChart ChartType="ArmsCandleVolume"
                     x:Name="financialChart"
                     SelectionMode="Series"
                     BindingX="date"
                     Binding="high,low,open,close,volume"
                     Grid.Row="1">
    <c1:FinancialSeries SeriesName="Series" />
</c1:C1FinancialChart>
```

Code

Visual Basic copyCode

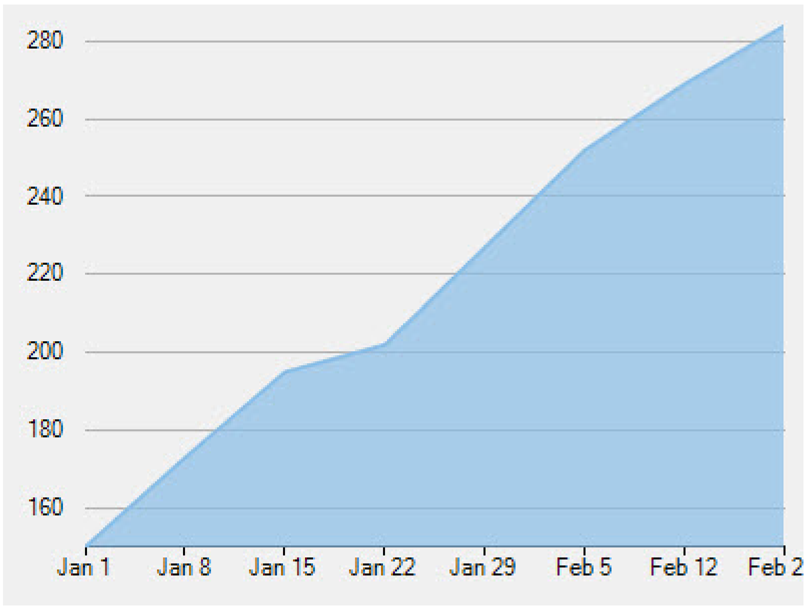
```
' set the financial chart type
financialChart.ChartType = FinancialChartType.ArmsCandleVolume
```

C# copyCode

```
// set the financial chart type
financialChart.ChartType = FinancialChartType.ArmsCandleVolume;
```

Area Chart

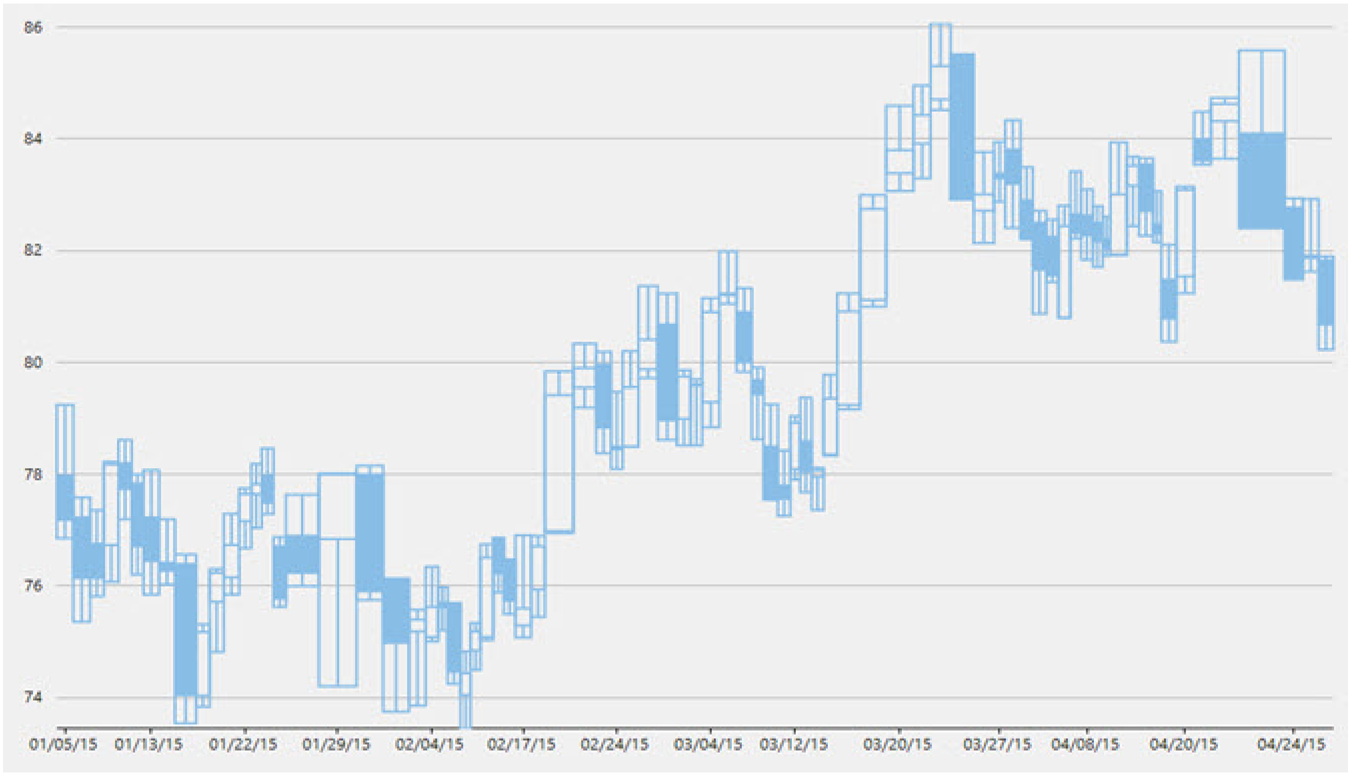
The Area Chart draws each series as connected points of data and the area below the connected points is filled with color to denote volume. Each new series is drawn on top of the preceding series. The series can either be drawn independently or stacked. These charts are commonly used to show trends between associated attributes over time.



Back to Top

ArmsCandleVolume Chart

Created by Richard Arms, the ArmsCandleVolume Chart is a combination of EquiVolume and CandleVolume chart types. The data for this chart type can be defined using the FinancialChart or FinancialSeries Binding property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty, volumeProperty". This chart type can only be used at the FinancialChart level, and should not be applied on FinancialSeries objects. Only one set of volume data is currently supported per FinancialChart.



Back to Top

Candlestick Chart

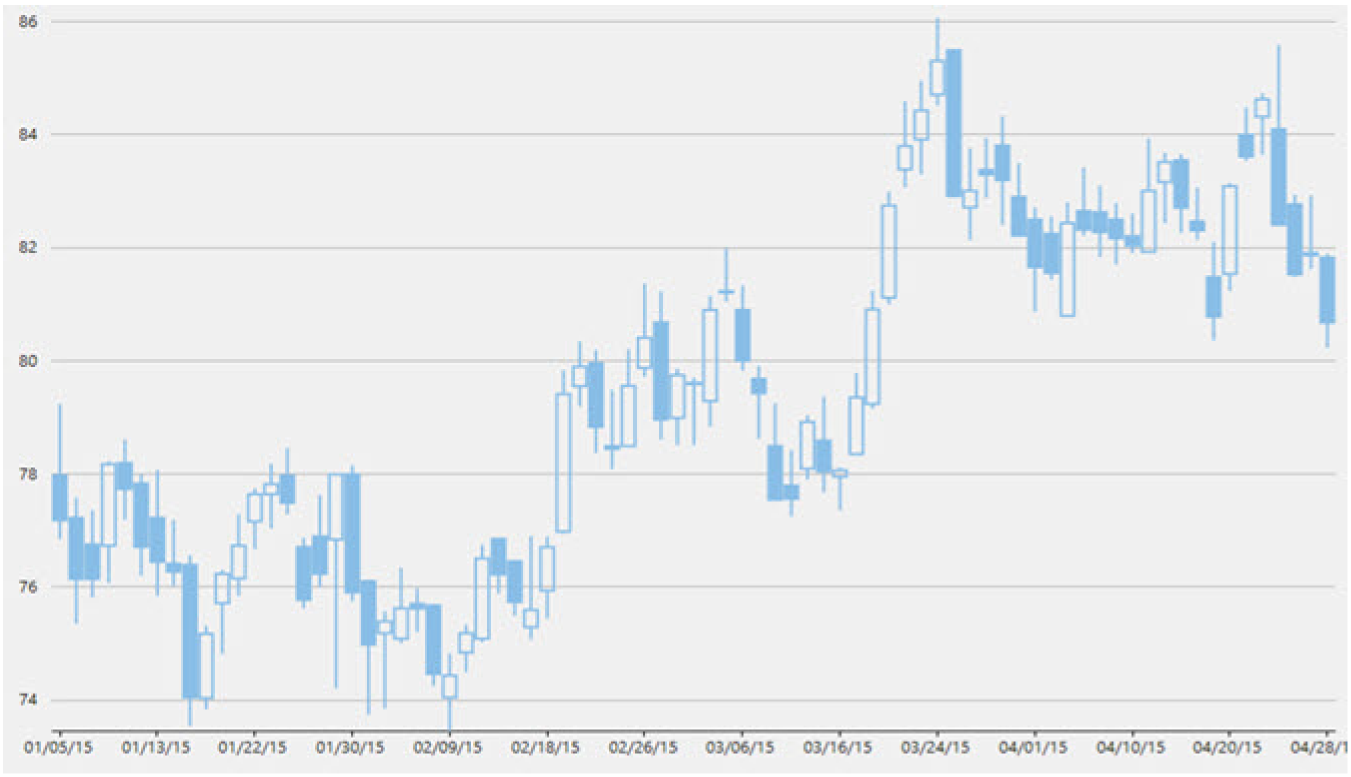
The Candlestick Chart is a special type of HiLoOpenClose Chart that shows the opening, closing, high, and low prices of a given stock. It integrates Bar and Line charts to depict a range of values over time. It consists of visual elements known as candles that are further comprised of three elements: body, wick, and tail.

- The **body** represents the opening and the closing value, while the **wick** and the **tail** represent the highest and the lowest value respectively.
- A **hollow body** indicates a rising stock price (the closing value is greater than the opening value).
- A **filled body** indicates a falling stock price (the opening value is greater than the closing value).

The size of the wick line is determined by the High and Low values, while the size of the bar is determined by the Open and Close values. The bar is displayed using different colors, depending on whether the close value is higher or lower than the open value. The data for this chart type can be defined using the FinancialChart or FinancialSeries binding property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty".

In a Candlestick, there are five values for each data point in the series.

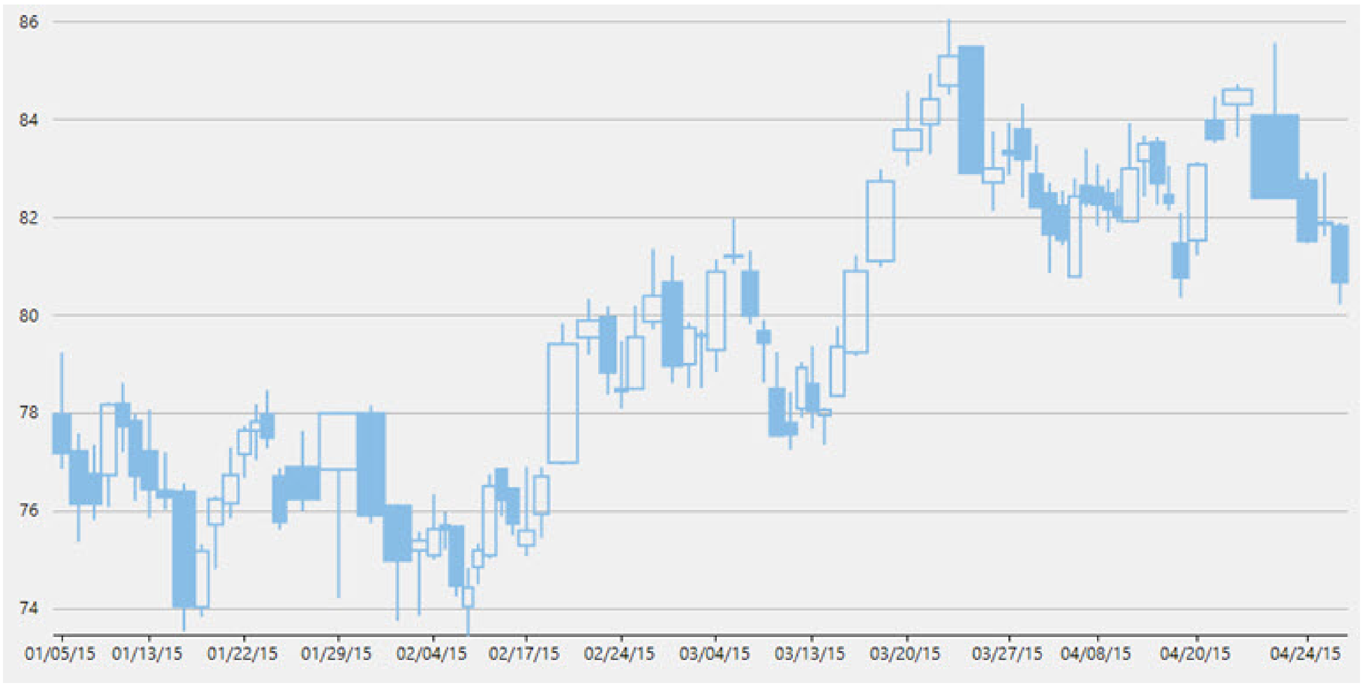
- **x**: Determines the date position along the x axis.
- **high**: Determines the highest price for the day, and plots it as the top of the candle along the y axis.
- **low**: Determines the lowest price for the day, and plots it as the bottom of the candle along the y axis.
- **open**: Determines the opening price for the day.
- **close**: Determines the closing price for the day.



Back to Top

CandleVolume Chart

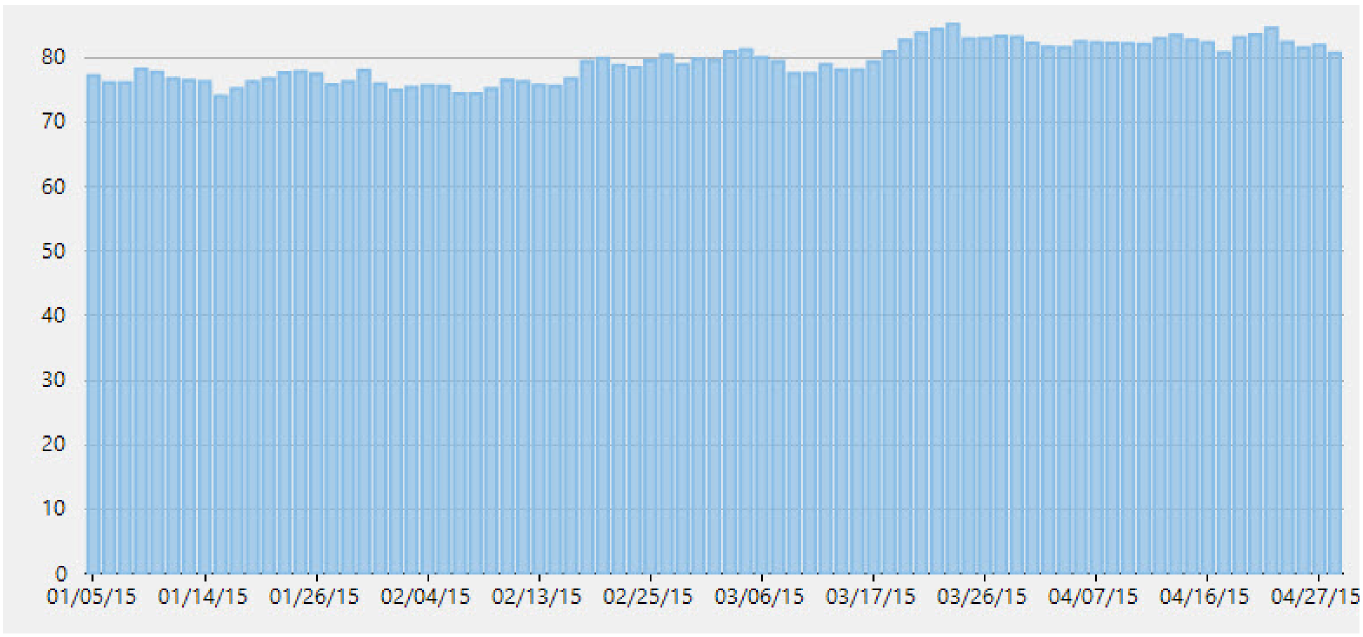
The CandleVolume Chart is identical to the standard Candlestick Chart except that the width of each bar in CandleVolume charts is determined by the Volume value. The data for this chart type can be defined using the FinancialChart or FinancialSeries binding property as a comma separated value in the following format: "highProperty, lowProperty, openProperty, closeProperty, volumeProperty".



Back to Top

Column Chart

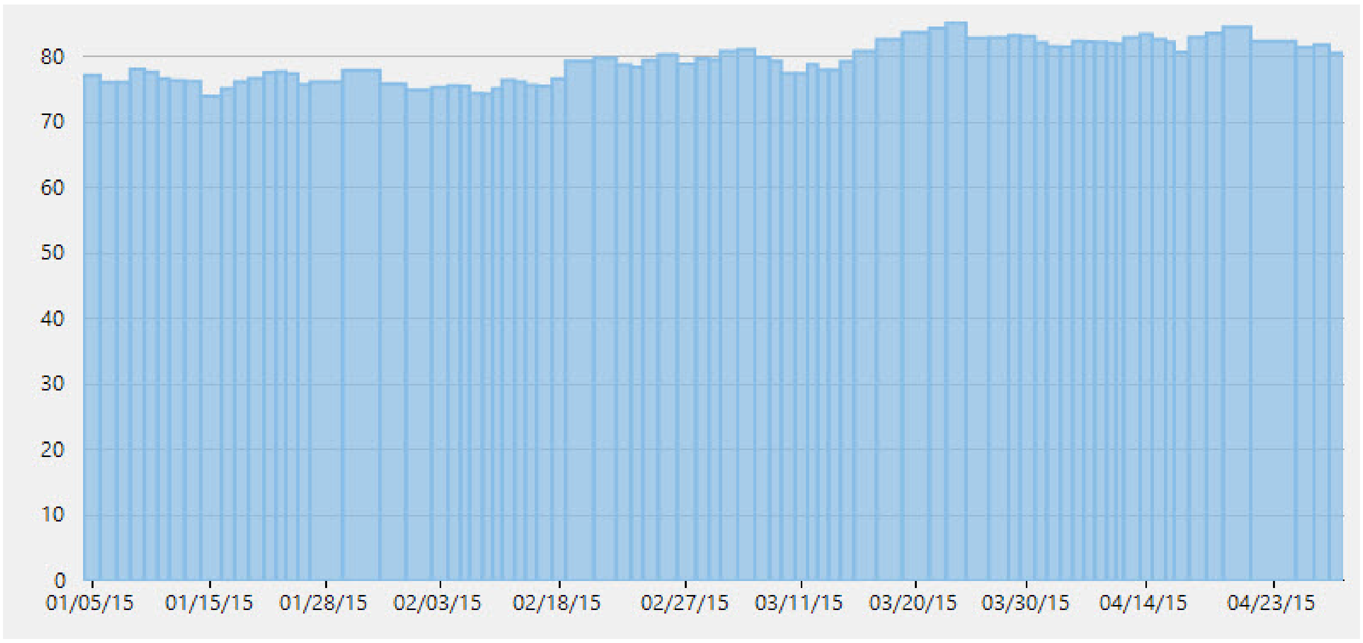
The Column Chart shows each series in the form of bars and enables you to compare values of items across different categories. It displays values of one or more items as vertical bars against Y-axis and arranges items or categories on X-axis.



Back to Top

ColumnVolume Chart

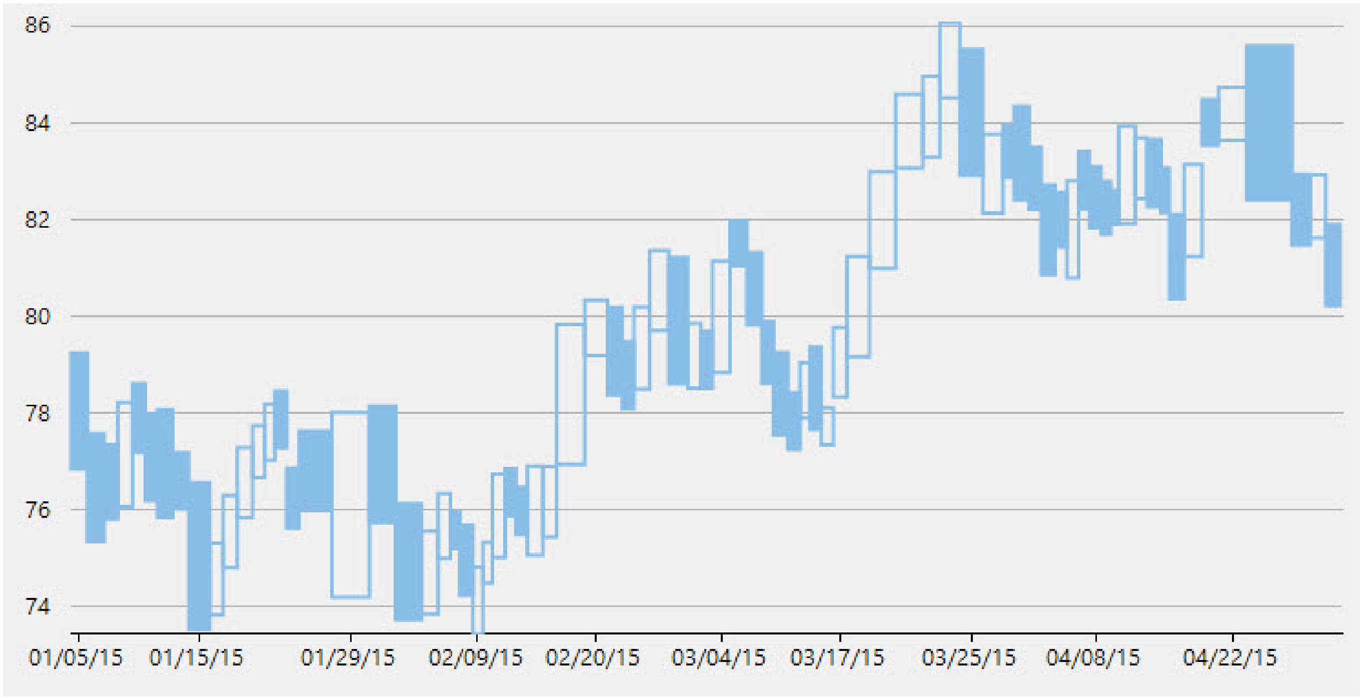
The ColumnVolume Chart is identical to the standard Column Chart except that the width of each bar is determined by the Volume value. The data for this chart type can be defined using the FinancialChart or FinancialSeries binding property as a comma separated value in the following format: "yProperty, volumeProperty".



Back to Top

EquiVolume Chart

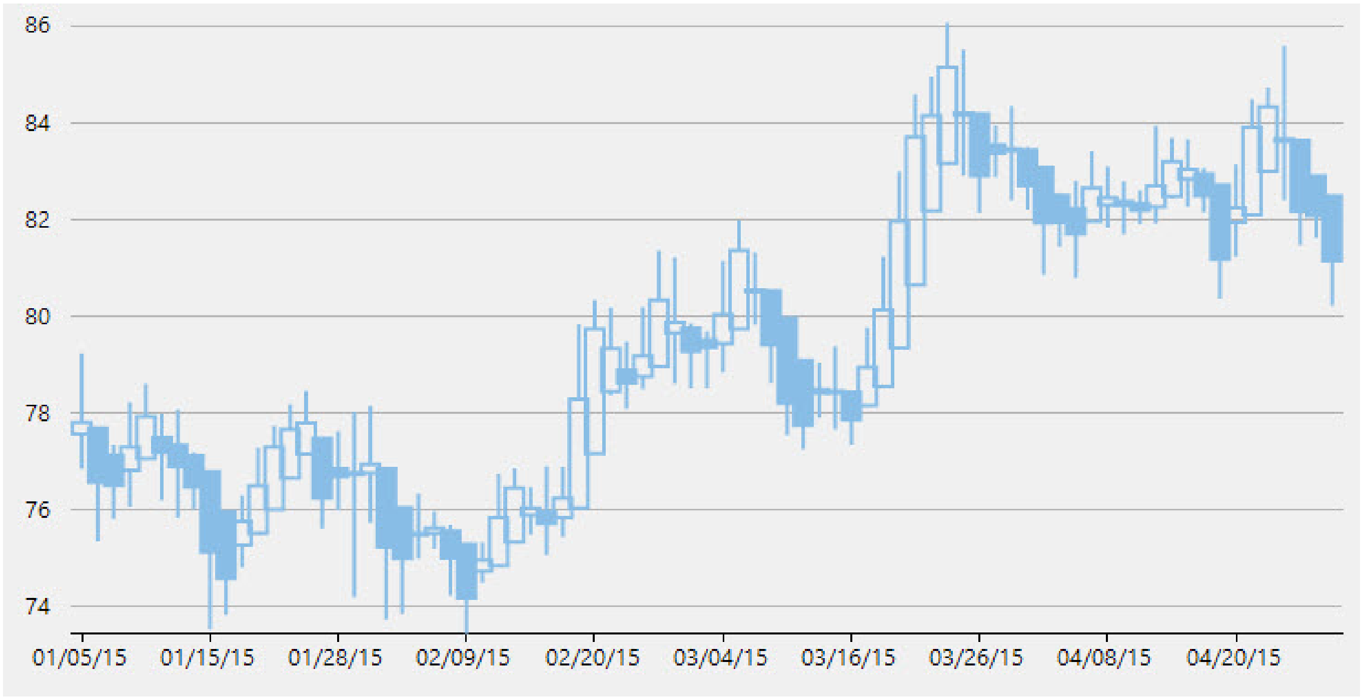
EquiVolume charts are similar to candlestick charts, but the candlesticks in these charts are replaced with rectangular boxes of varying width (and no wicks). An EquiVolume box includes high and low price components with a third dimension, Volume that determines the width of each box. Color represents whether the close number is higher or lower than the previous box's close.



Back to Top

HeikinAshi Chart

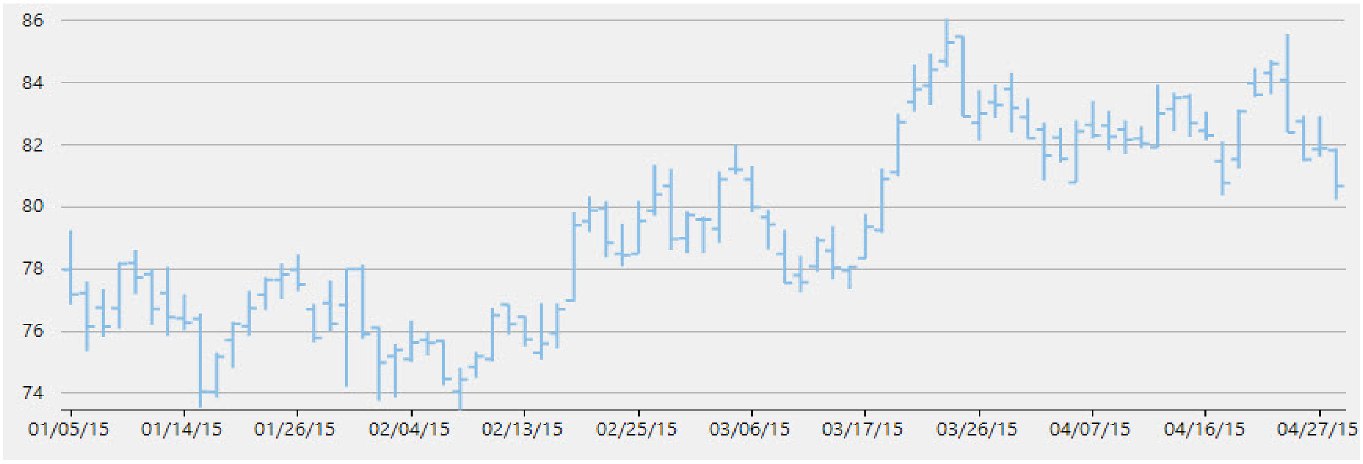
HeikinAshi charts are a variation of Japanese candlestick charts that were designed to remove noise from candlesticks and behave much like a moving average. These charts can be used to identify trends, potential reversal points, and other technical analysis patterns.



Back to Top

HighLowOpenClose Chart

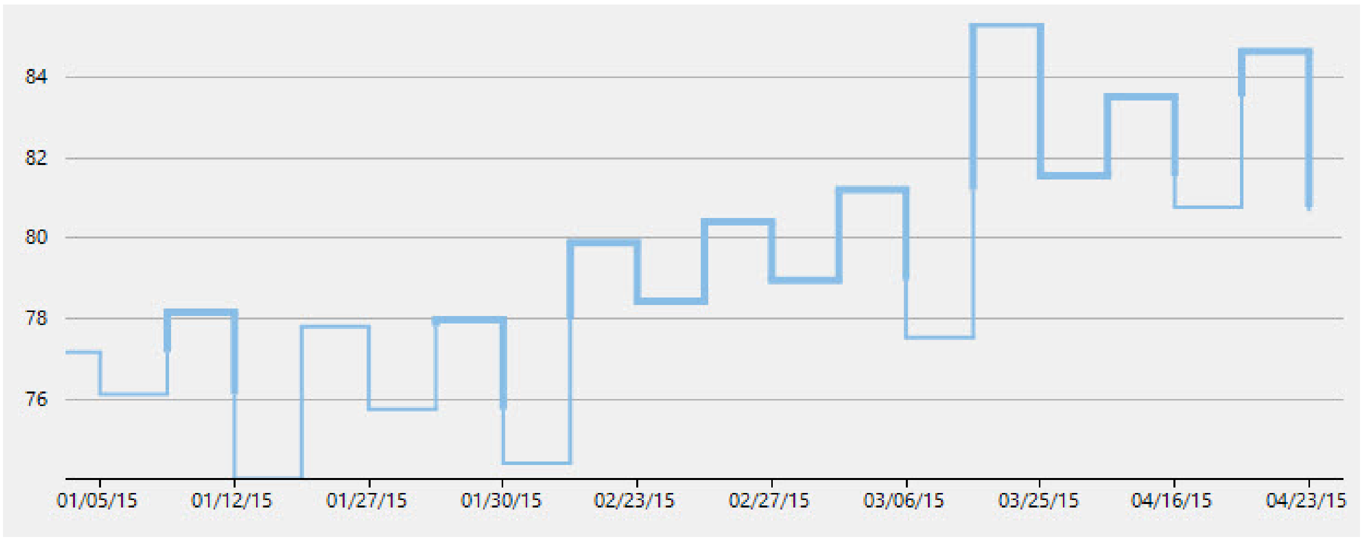
The HiLoOpenClose Chart combines four independent values to supply high, low, open, and close data for a point in a series. It displays the same information as the Candlestick Chart except that opening values are displayed using lines to the left, while lines to the right indicate closing values.



Back to Top

Kagi Chart

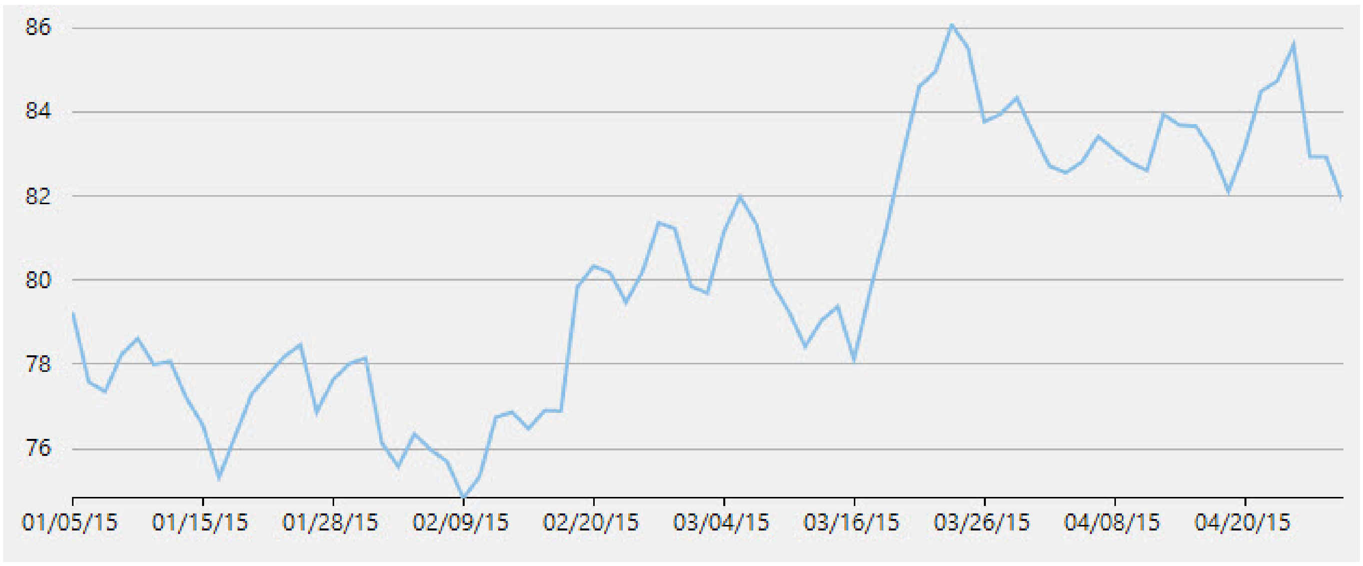
A Kagi chart displays supply and demand trends using a sequence of linked vertical lines. The thickness and direction of the lines vary depending on the price movement. If closing prices go in the direction of the previous Kagi line, then that Kagi line is extended. However, if the closing price reverses by the preset reversal amount, a new Kagi line is charted in the next column in the opposite direction. Thin lines indicate that the price breaks the previous low (supply) while thick lines indicate that the price breaks the previous high (demand).



Back to Top

Line Chart

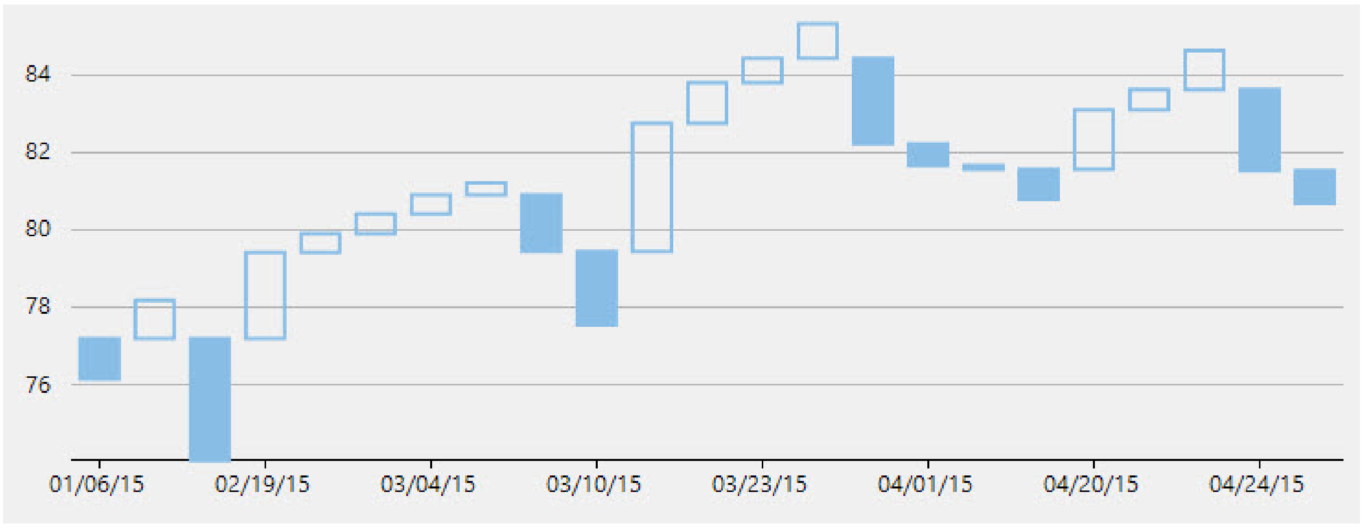
The Line Chart displays trends over a period of time by connecting different data points in a series with a straight line. It treats the input as categorical information that is evenly spaced along the X-axis.



Back to Top

LineBreak Chart

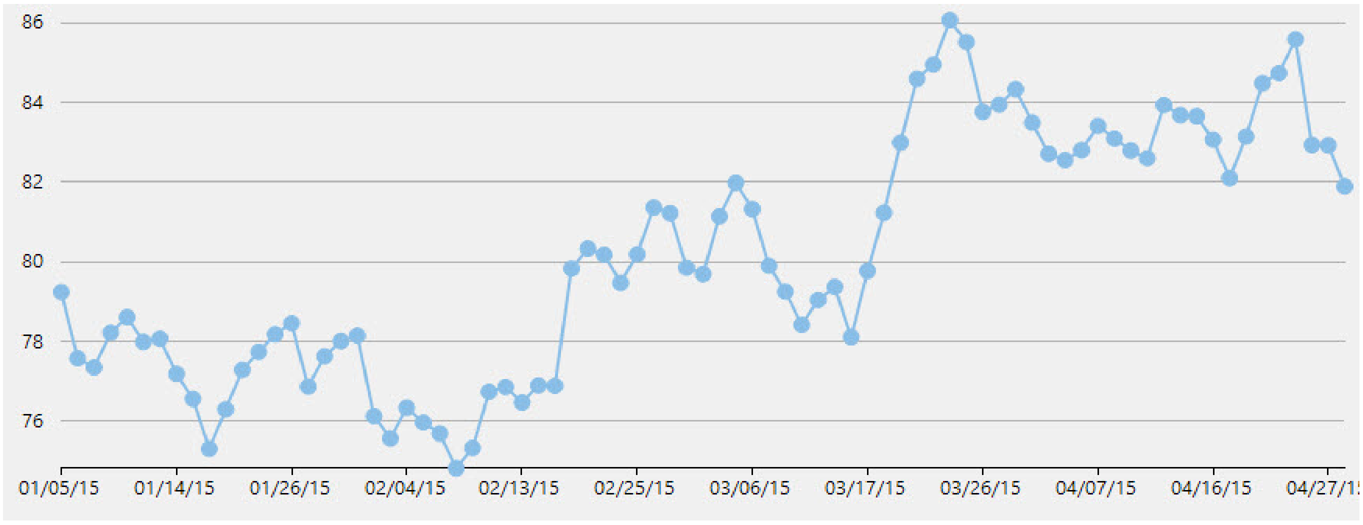
The Line Break Chart uses vertical boxes or lines to illustrate the price changes of an asset or market. Movements are depicted with box colors and styles; movements that continue the trend of the previous box are colored similarly while movements that trend oppositely are indicated with a different color and/or style. The opposite trend is only drawn if its value exceeds the extreme value of the previous n number of boxes or lines, which is determined by the newLineBreaks option.



Back to Top

LineSymbols Chart

The LineSymbols Chart is a combination of the Line Chart and the Scatter Chart. The chart displays trends in data at equal intervals and visualizes relationship between two variables related to the same event. It plots data points by using symbols and connects the data points by using straight lines.

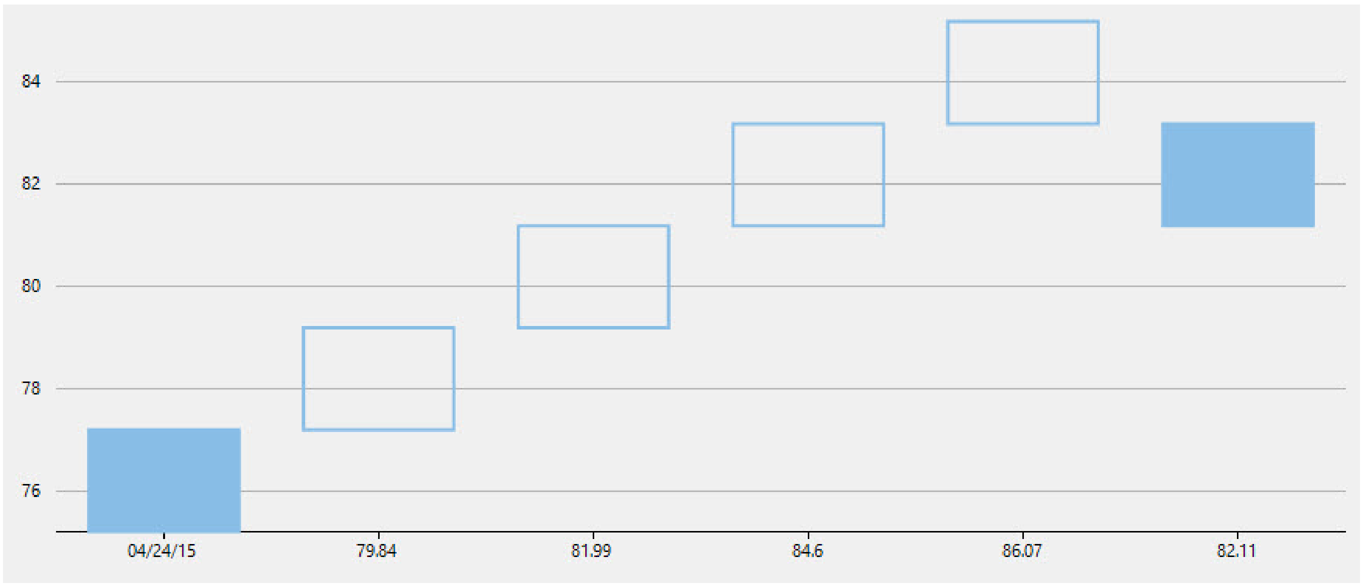


Back to Top

Renko Chart

s time and focuses on price changes that meet a specified amount.

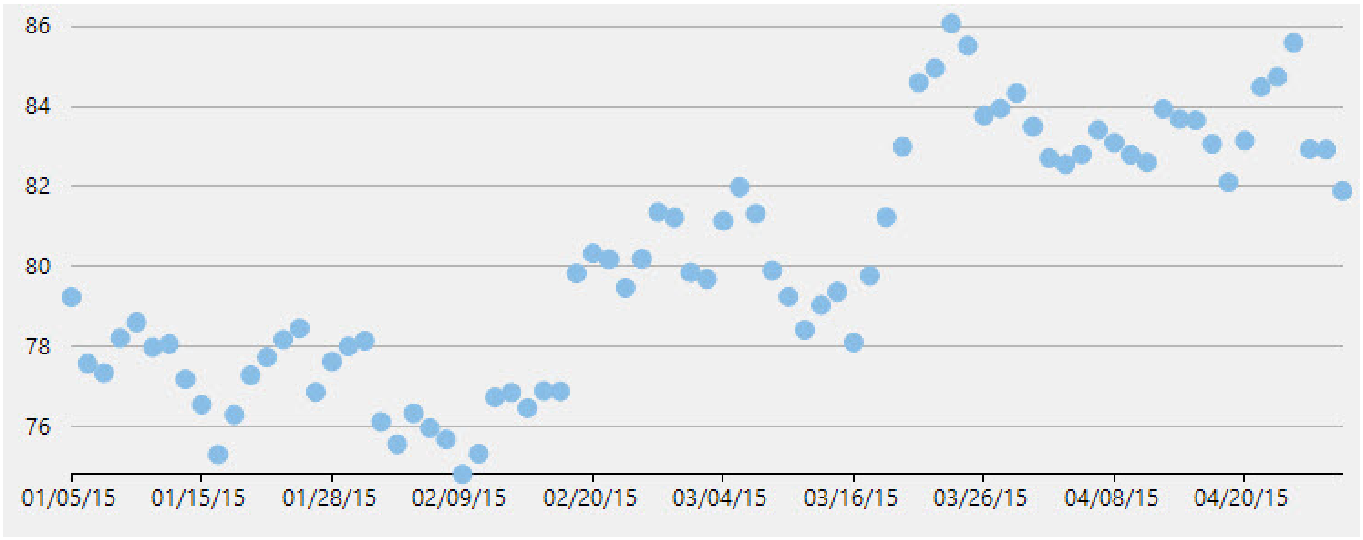
The Renko Chart ignores time and focuses on price changes that meet a specified amount. It displays the price movement by using bricks of uniform size. When a price moves to a greater or lesser value than the preset boxSize option required to draw a new brick, a new brick is drawn in the succeeding column. The change in box color and direction signifies a trend reversal.



Back to Top

Scatter Chart

The Scatter Chart, which is also known as the XY Chart, depicts relationship among items of different data series. In simple terms, it is a plot of X values and Y values along the two axes. The data points are not connected and can be customized using different symbols. This chart type is normally used to represent scientific data, and can highlight the deviation of assembled data from predicted data or result.



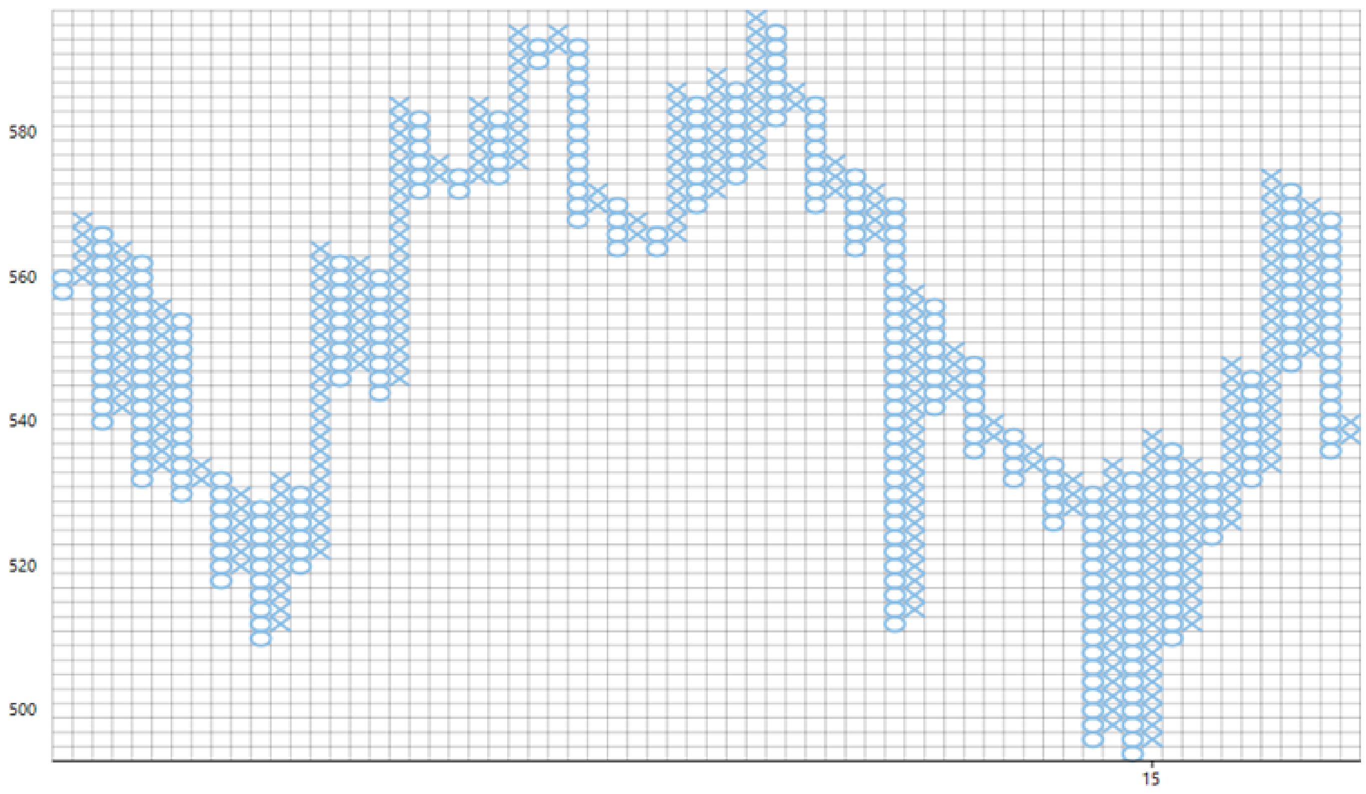
Back to Top

Point and Figure Chart

Point and Figure chart defines price trends through columns of stacked Xs or Os. Unlike time based charts, these charts focus on price movements alone without considering time. While Xs represent rising prices, Os represent falling prices.

You can specify box size, which is the value represented by each X and O. An increase in price, by the amount equal to box size, is recorded by an X. Further increase is recorded by another X stacked above it and so on. Whereas to show an opposite trend, the fall in price by the magnitude of box size is recorded by an O in the new column. It is tedious to record every price change, therefore you can set reversal amount to specify the minimum Xs or Os in a column before a price reversal is recorded. Price reversals occurring before the specified reversal amount are ignored.

FinancialChart supports Traditional, Fixed and Dynamic scaling in Point and Figure chart. While, Traditional scaling uses predefined table of price ranges provided by ChartCraft to determine box size, Fixed scaling allows end users to specify a box size and Dynamic scaling uses the calculated ATR value as box size.



Back to Top

Analytics

FinancialChart for WPF offers a number of analytics that help you analyze financial data systematically and effectively. These analytics include trendlines, moving average, and various technical indicators that allow you to analyze problems of prediction, examine overall data, and forecast the assets' market direction respectively.

Click the following links to know more about FinancialChart analytics:

- Trendlines
- Moving Average
- Indicators
- Overlays
- Fibonacci Tool

Trendlines

Trendlines are a crucial tool in technical analysis for identifying and confirming trends. A straight line connecting two or more price points, a trendline can act as a line of resistance or support in future. Basically, trendlines are used to depict trends in data and to examine problems of prediction. Commonly used with financial charts, these lines can be used with diversified technical analysis charts, for instance MACD (moving average convergence/divergence) that is a trading indicator used in technical analysis of stock prices, or RSI (relative strength index) that is a technical indicator used in the analysis of financial markets.

The supported FitTypes can be set using the FitType property that accepts the following values from the FitType enumeration. Each trend type is drawn based on the calculation formula of its type.

Type	Description
Average X	Calculates the average value of X from the chart data and draws a trendline.
Average Y	Calculates the average value of Y from the chart data and draws a trendline.
Exponential	A curved line that is convenient to use when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline if your data contains zero or negative values.
Fourier	A way to display a wave like function as a combination of simple sine waves. It is created by using the fourier series formula.
Linear	A linear trendline is a best-fit straight light. Your data is linear if the data point pattern resembles a line, and a linear trendline is a good fit if the R-squared value is at or near 1.
Logarithmic	A best fit curved line used for better visualization of data. Used when the rate of change in the data increases or decreases quickly and then levels out. It can also use positive and negative values.
Max X	Takes the maximum value of X from the chart and draws a trendline using it.
Max Y	Takes the maximum value of Y from the chart and draws a trendline using it.
Min X	Takes the minimum value of X from the chart and draws a trendline using it.
Min Y	Takes the minimum value of Y from the chart and draws a trendline using it

Polynomial	A twisted line that is used when data oscillates. It is useful for analyzing gains and losses over a large data set.
Power	A curved line that is best used with data sets that compare calculation that increase at a peculiar rate. For example, the acceleration of a vehicle at one-second intervals.

Create a Trendline instance, and set properties, such as FitType, Order, and SampleCount of the Trendline class. Add the trendline instance to the Series collection.

XAML

```
<c1:C1FinancialChart Binding="Sales"
                     BindingX="Date"
                     x:Name="financialChart"
                     ChartType="LineSymbols"
                     ItemsSource="{Binding DataContext.Data}"
                     HorizontalAlignment="Left"
                     Height="321"
                     VerticalAlignment="Top"
                     Width="620"
                     Margin="79,85,0,0">
    <c1:FinancialSeries AxisX="{x:Null}"
                        AxisY="{x:Null}"
                        Chart="{x:Null}"
                        SeriesName="{x:Null}">
    </c1:FinancialSeries>
    <c1:TrendLine FitType="Fourier"
                  Order="10"
                  SampleCount="150"
                  x:Name="trendLine" />
</c1:C1FinancialChart>
```

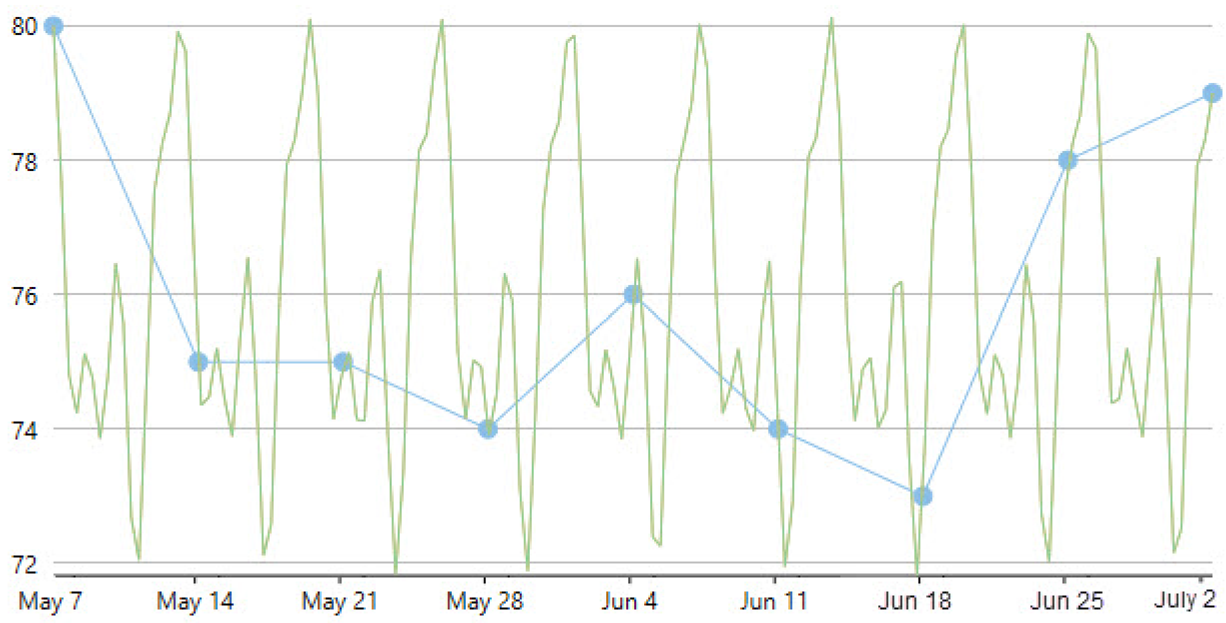
Code

Visual Basic copyCode

```
' create a Trendline instance
Dim trendline As New C1.WPF.Chart.TrendLine()

' set the properties of the Trendline instance
trendline.FitType = C1.Chart.FitType.Fourier
trendline.SampleCount = 150
trendline.Order = 10

' add the Trendline instance to the Series collection
financialChart.Series.Add(trendline)
```

C# copyCode

```
// create a Trendline instance
C1.WPF.Chart.TrendLine trendline = new C1.WPF.Chart.TrendLine();

// set the properties of the Trendline instance
trendline.FitType = C1.Chart.FitType.Fourier;
trendline.SampleCount = 150;
trendline.Order = 10;

// add the Trendline instance to the Series collection
financialChart.Series.Add(trendline);
```



Moving Average

Moving Average is a moving average trendline used in financial charts. It analyzes data points by creating series of averages of various subsets of the complete data set.

In FinancialChart, you can create an instance of the MovingAverage class, and set the Type property to any of the following values from the MovingAverageType enumeration:

- **Exponential**: Weighted average of the last n values, where the weightage decreases exponentially with each previous value.
- **Simple**: An average of the last n values.
- **Triangular**: Weighted average of the last n values, whose result is equivalent to a double smoothed simple moving average.
- **Weighted**: Weighted average of the last n values, where the weightage decreases by 1 with each previous value.

You can set the ChartType property to specify the chart type for the moving average. The property accepts values from the FinancialChartType enumeration. For more details on chart types, refer to Financial Chart Types.

In addition, you can use the Period property to specify the period of the moving average. Once you have set the properties, add the moving average to the Series collection.

XAML

```
<c1:C1FinancialChart Binding="Sales"
                     BindingX="Date"
                     x:Name="financialChart"
```

```
                        ItemsSource="{Binding DataContext.Data}"
                        HorizontalAlignment="Left"
                        Height="321"
                        VerticalAlignment="Top"
                        Width="620"
                        Margin="79,85,0,0">
    <c1:FinancialSeries ChartType="LineSymbols"
                           AxisX="{x:Null}"
                           AxisY="{x:Null}"
                           Chart="{x:Null}"
                           SeriesName="{x:Null}">
    </c1:FinancialSeries>
    <c1:MovingAverage x:Name="ma"
                         Type="Weighted"
                         ChartType="Line"
                         Period="2"/>
</c1:C1FinancialChart>
```

Code

Visual Basic copyCode

```vb
' create an instance of the Moving Average class
Dim ma As New C1.WPF.Chart.Finance.MovingAverage()

' set the properties for the Moving Average instance
ma.Type = C1.Chart.MovingAverageType.Weighted
ma.ChartType = C1.Chart.Finance.FinancialChartType.Line
ma.Period = 2

' Add the Moving Average instance to the Series collection
financialChart.Series.Add(ma)
```

C# copyCode

```csharp
// create an instance of the Moving Average class
C1.WPF.Chart.Finance.MovingAverage ma = new C1.WPF.Chart.Finance.MovingAverage();

// set the properties for the Moving Average instance
ma.Type = C1.Chart.MovingAverageType.Weighted;
ma.ChartType = C1.Chart.Finance.FinancialChartType.Line;
ma.Period = 2;

// Add the Moving Average instance to the Series collection
financialChart.Series.Add(ma);
```

Indicators

A technical indicator is a set of derived data that is calculated by applying one or more formulas to the original set of data. Technical indicators are generally used to forecast the asset's market direction and generally plotted separately from the original data since the Y-axis scales differ.

WPF Edition supports technical indicators for its FinancialChart control to be easily used in financial applications. These financial indicators are plotted as chart patterns and form a basis for technical analysis.

Note that the indicators are generally plotted separately from original price or volume data, as Y axis scales for technical indicators differ from that of price or volume chart data.

The following sections discuss various financial chart indicators that FinanicalChart for WPF supports:

- Average True Range
- Relative Strength Index
- Commodity Channel Index
- Williams %R
- Stochastic
- Moving Average Convergence Divergence

Average True Range

Average True Range (ATR) is a technical indicator for measuring the volatility of an asset. It does not provide an indication of the price trend, but of the degree of the price volatility. It is typically based on 14 periods, and could be calculated intra daily, daily, weekly or monthly basis. Stocks having high volatility will have a higher ATR, while low volatility stocks will have a lower ATR.

FinancialChart also enables you to fetch the calculated ATR values at run-time using GetValues() method. This can help in creating alerts in application or maintaining logs while working with dynamic data.

The following code snippet creates an instance of the ATR class to use Average True Indicator. Also, the sample uses a class **DataService.cs** to get data for the financial chart.

- **DataService.vb**

```
Public Class DataService
    Private _companies As New List(Of Company)()
    Private _cache As New Dictionary(Of String, List(Of Quote))()

    Private Sub New()
        _companies.Add(New Company() With {
            Key.Symbol = "box",
            Key.Name = "Box Inc"
        })
        _companies.Add(New Company() With {
            Key.Symbol = "fb",
            Key.Name = "Facebook"
        })
    End Sub

    Public Function GetCompanies() As List(Of Company)
        Return _companies
    End Function

    Public Function GetSymbolData(symbol As String) As List(Of Quote)
        If Not _cache.Keys.Contains(symbol) Then
            Dim path As String = String.Format("FinancialChartExplorer.Resources.{0}.json", symbol)
            Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
            Dim ser = New DataContractJsonSerializer(GetType(Quote()))
            Dim data = DirectCast(ser.ReadObject(stream), Quote())
            _cache.Add(symbol, data.ToList())
        End If

        Return _cache(symbol)
    End Function

    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    List<Company> _companies = new List<Company>();
    Dictionary<string, List<Quote>> _cache = new Dictionary<string, List<Quote>>();

    private DataService()
    {
        _companies.Add(new Company() { Symbol = "box", Name = "Box Inc" });
        _companies.Add(new Company() { Symbol = "fb", Name = "Facebook" });
    }

    public List<Company> GetCompanies()
    {
        return _companies;
    }

    public List<Quote> GetSymbolData(string symbol)
    {
        if (!_cache.Keys.Contains(symbol))
        {
            string path = string.Format("FinancialChartExplorer.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            _cache.Add(symbol, data.ToList());
        }

        return _cache[symbol];
    }

    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

- **Visual Basic**

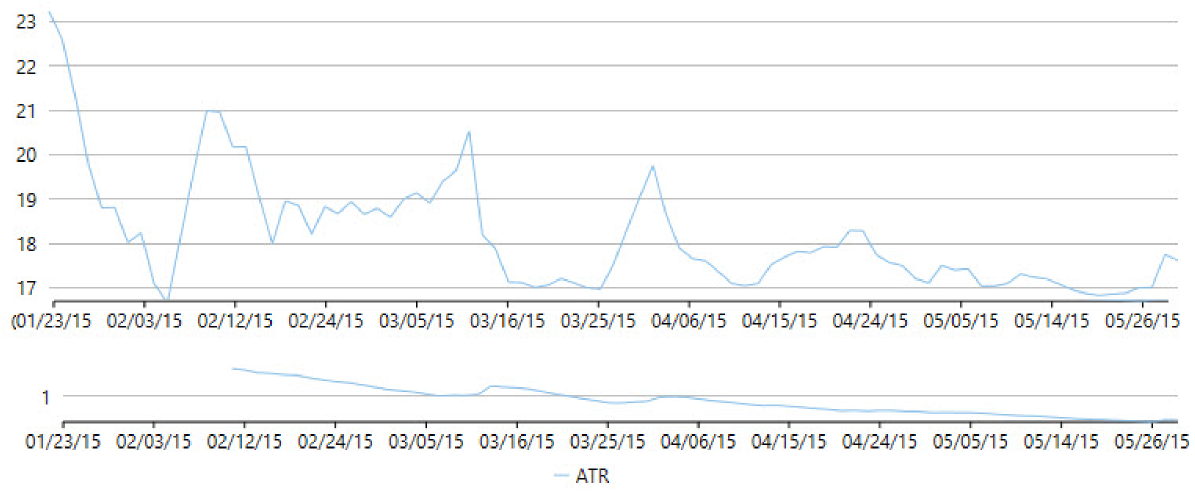
```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = DataService.GetService()
    Private atr As New ATR() With {
        Key.SeriesName = "ATR"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Average True Range"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
```

```
            ser = atr
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    ATR atr = new ATR() { SeriesName = "ATR" };

    public Indicators()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Average True Range",
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = atr;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
```

```
            {
                indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
                indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
            }
        }
    }
```



Relative Strength Index

Relative Strength Index (RSI) indicator for FinancialChart is a momentum oscillator, which measures velocity and magnitude of price movements. It compares the upward movements in closing price of an asset to the downward movements over a trading period, and intends to determine strength or weakness of a stock. It fluctuates between 0 and 100. The stocks with strong positive changes have a higher RSI than the stocks with strong negative changes.

It finds application in comparing the magnitude of recent gains to recent losses, to determine the overbought and oversold conditions of an asset. Stocks are considered overbought when RSI is above 70, and oversold when below 30.

FinancialChart also enables you to fetch the calculated RSI values at run-time using GetValues() method. This can help in creating alerts in application or maintaining logs while working with dynamic data.

Notice that the given code snippet uses a class **DataService.cs.** To see the code, refer to Average True Range. In addition, the sample creates an instance of the RSI class to work with Relative Strength Index.

- **DataService.vb**

```
Public Class DataService
    Private _companies As New List(Of Company)()
    Private _cache As New Dictionary(Of String, List(Of Quote))()

    Private Sub New()
        _companies.Add(New Company() With {
            Key.Symbol = "box",
            Key.Name = "Box Inc"
        })
        _companies.Add(New Company() With {
            Key.Symbol = "fb",
            Key.Name = "Facebook"
        })
    End Sub

    Public Function GetCompanies() As List(Of Company)
        Return _companies
    End Function

    Public Function GetSymbolData(symbol As String) As List(Of Quote)
        If Not _cache.Keys.Contains(symbol) Then
            Dim path As String = String.Format("FinancialChartExplorer.Resources.{0}.json", symbol)
            Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
            Dim ser = New DataContractJsonSerializer(GetType(Quote()))
            Dim data = DirectCast(ser.ReadObject(stream), Quote())
            _cache.Add(symbol, data.ToList())
        End If

        Return _cache(symbol)
    End Function

    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

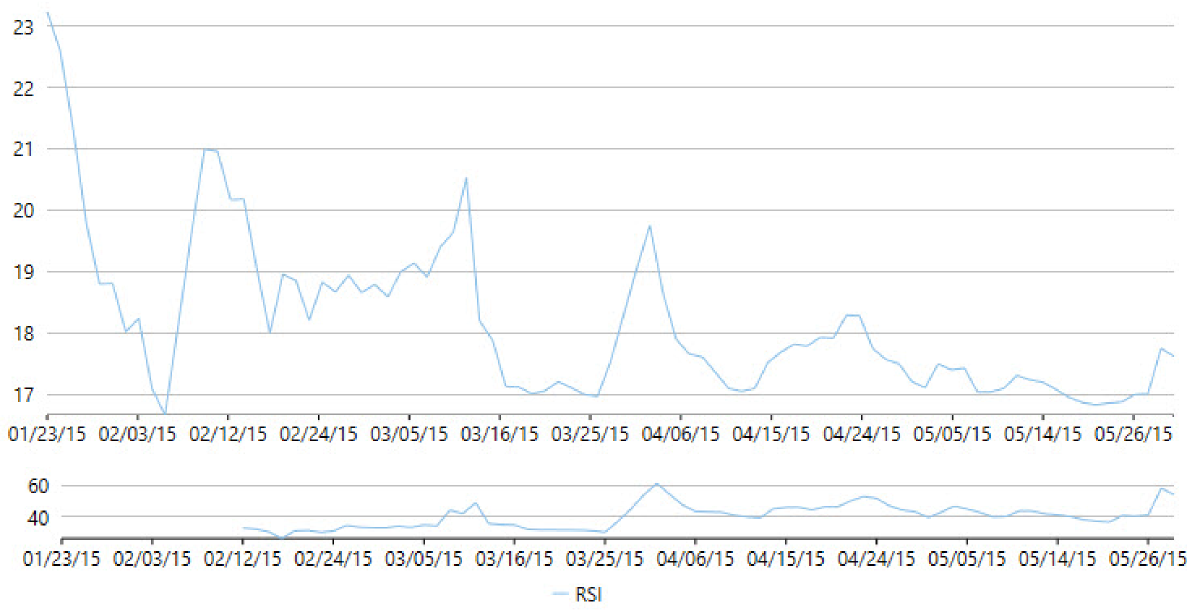
```
public class DataService
{
    List<Company> _companies = new List<Company>();
    Dictionary<string, List<Quote>> _cache = new Dictionary<string, List<Quote>>();

    private DataService()
    {
        _companies.Add(new Company() { Symbol = "box", Name = "Box Inc" });
        _companies.Add(new Company() { Symbol = "fb", Name = "Facebook" });
    }

    public List<Company> GetCompanies()
    {
        return _companies;
    }

    public List<Quote> GetSymbolData(string symbol)
    {
        if (!_cache.Keys.Contains(symbol))
        {
            string path = string.Format("FinancialChartExplorer.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            _cache.Add(symbol, data.ToList());
        }

        return _cache[symbol];
    }

    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

- **Visual Basic**

```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = DataService.GetService()
    Private rsi As New RSI() With {
        Key.SeriesName = "RSI"
    }

    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Relative Strength Index"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = rsi
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    RSI rsi = new RSI() { SeriesName = "RSI" };

    public Indicators()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Relative Strength Index",
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = rsi;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



Commodity Channel Index

Commodity Channel Index (CCI) indicator is an oscillator that measures an asset's current price level relative to an average price level over a specified period of time. It is used to determine a new trend or to warn about extreme conditions.

In FinancialChart, you need to use a CCI object to work with Commodity Channel Index. FinancialChart also enables you to fetch the calculated CCI values at run-time using GetValues() method. This can help in creating alerts in application or maintaining logs while working with dynamic data.

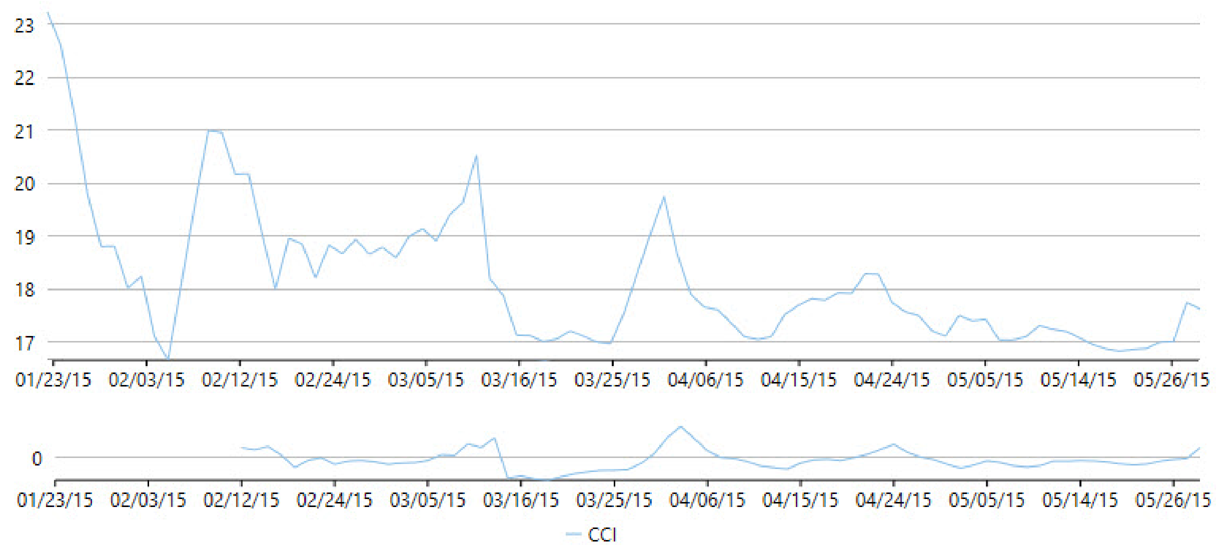
See the following code snippet that demonstrates how you can use CCI indicator. The code snippet uses a class **DataService.cs** whose code can be seen by referring to Average True Range.

- **Visual Basic**

```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = dataService.GetService()
    Private cci As New CCI() With {
        Key.SeriesName = "CCI"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Commodity Channel Index"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = cci
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    CCI cci = new CCI() { SeriesName = "CCI" };

    public Indicators()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Commodity Channel Index",
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = cci;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



Williams %R

Williams %R indicator for the FinancialChart is a momentum indicator, which compares the current asset price to the highest price over the look back period. Its look-back is typically 14 periods. The indicator fluctuates between 0 and -100. It is the inverse of a fast Stochastic Oscillator.

While the Williams %R displays the level of a stock's close relative to the highest high for the look-back period, the Stochastic Oscillator shows the level of a stock's close relative to the lowest low. Both the indicators show same lines, however scaling is different. It finds application in determining Overbought/Oversold levels, providing buy and sell signals, and momentum confirmations.

To work with WilliamsR indicator, you need to create an instance of WilliamsR class. FinancialChart also enables you to fetch the calculated WilliamsR values at run-time using GetValues() method. This can help in creating alerts in application or maintaining logs while working with dynamic data.

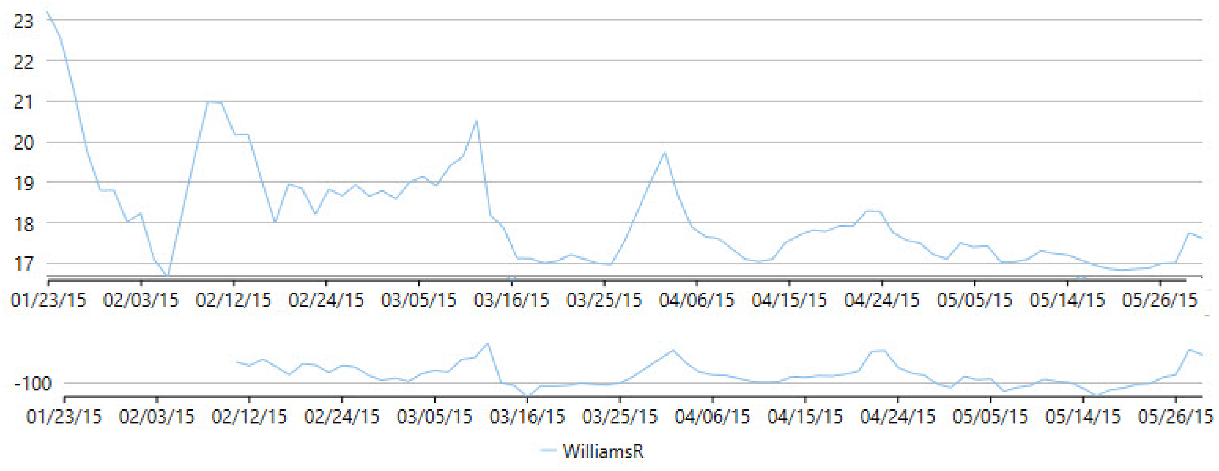
The following code snippet demonstrates how to use the indicator.

- **Visual Basic**

```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = dataService.GetService()
    Private wr As New WilliamsR() With {
        Key.SeriesName = "WilliamsR"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Williams %R"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = wr
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    WilliamsR wr = new WilliamsR() { SeriesName = "WilliamsR" };

    public Indicators()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Williams %R"
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = wr;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



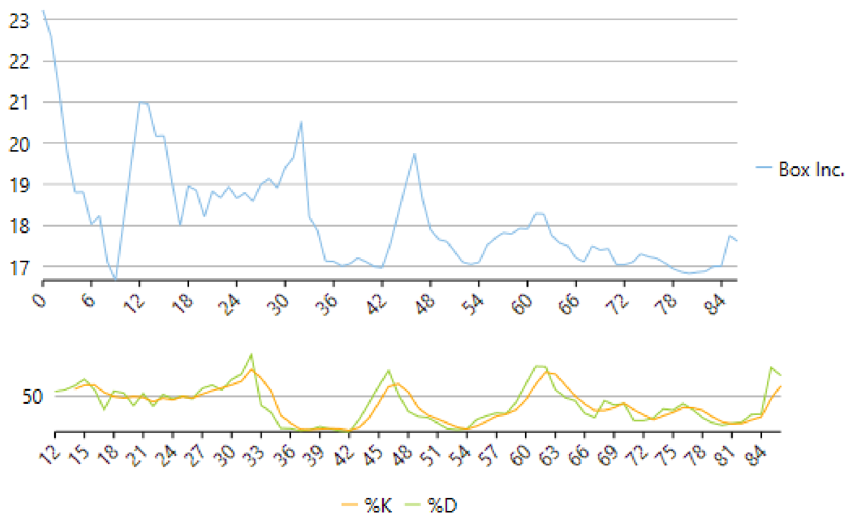
Stochastic

Stochastic indicator is a momentum indicator to foreshadow price turning points. It compares the closing price of a financial instrument with its range of prices over a period of time. It can be used to anticipate future reversals by identifying bull and bear set-ups.

Stochastic Oscillator indicator is measured with K line and D line. The D line is followed closely to indicate any major signals in the FinancialChart. To create a slow stochastic oscillator, SmoothingPeriod is set to 3, while a SmoothingPeriod value of 2 creates full stochastic oscillator. To create to a fast stochastic oscillator, the SmoothingPeriod is set to an integer value of 1.

To use stochastic indicator in FinancialChart, add a FinancialChart control to your application and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart. The Stochastic class exposes KPeriod (takes integer value to calculate price range over the specified period), DPeriod (takes integer value to calculate the Moving Average of K line), and Smoothing period (takes integer value to create fast, full or slow oscillator) properties. Based on the values of these properties, data points for Stochastic indicator are calculated and plotted on FinancialChart. KLineStyle and DLineStyle properties can be utilized to change the appearance of series.

FinancialChart also enables you to fetch the calculated D values, D x values, K values, and K x values at run-time. This can help in creating alerts in application or maintaining logs while working with dynamic data.



The following example considers stock data for a company Box Inc. over a period of time and plots its Stochastic Oscillator Indicator apart from the volume chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

PriceChart.xaml copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:StochasticInd"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="StochasticInd.MainWindow"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             BindingX="Date"
                             Binding="Close"
                             ChartType="Line"
                             ItemsSource="{Binding Data}"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
                             Margin="10,29,10,177"
                             Rendered="OnFinancialChartRendered">

            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                SeriesName="Box Inc." />
            <c1:Fibonacci Binding="High,Low,Open,Close"
```

```
                      ChartType="Line"
                      LabelPosition="Bottom"
                      Uptrend="True"
                      Visibility="Plot" >
                <c1:Fibonacci.Style>
                    <c1:ChartStyle Fill="Red"
                                   Stroke="Red"
                                   StrokeThickness="0.5"
                                   FontSize="10"/>
                </c1:Fibonacci.Style>
            </c1:Fibonacci>
            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
```

IndicatorChart.xaml copyCode

```
        <c1:Stochastic x:Name="stochastic"
                       SeriesName="%K,%D"
                       DPeriod="3"
                       KPeriod="13"
                       SmoothingPeriod="1">
            <c1:Stochastic.DLineStyle>
                <c1:ChartStyle Stroke="Orange" />
            </c1:Stochastic.DLineStyle>
            <c1:Stochastic.KLineStyle>
                <c1:ChartStyle Stroke="YellowGreen" />
            </c1:Stochastic.KLineStyle>
        </c1:Stochastic>
        <c1:C1FinancialChart.AxisX>
            <c1:Axis LabelAngle="45" MajorUnit="3"/>
        </c1:C1FinancialChart.AxisX>
    </c1:C1FinancialChart>
</Grid>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "Indicator.Resources.box.json"
        'Replace Indicator by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "StochasticInd.Resources.box.json";
        //Replace StochasticInd by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double
    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for MainWindow.xaml
Partial Public Class MainWindow
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
        indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
    End Sub
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// Interaction logic for MainWindow.xaml
public partial class MainWindow : Window
{
    DataService dataService = DataService.GetService();
    public MainWindow()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }

    private void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
        indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
    }
 }
```

Back to Top

Moving Average Convergence Divergence

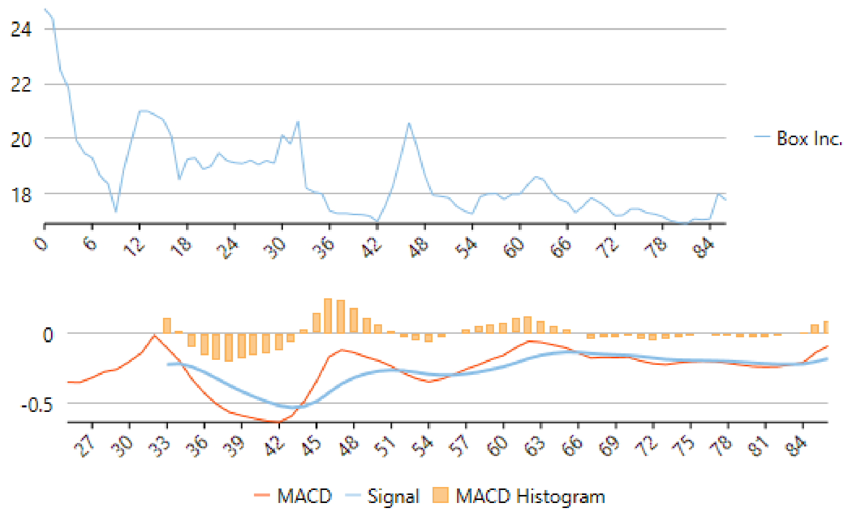
Moving Average Convergence Divergence (MACD) indicator for FinancialChart is a trend-following momentum indicator. It reveals changes in the strength, direction, duration and momentum of price of an asset. The indicator is efficient in helping users spot short-term price momentum.

MACD momentum oscillator displays a relationship between 26 day exponential moving average and 12 day exponential moving average. As the two moving averages converge, cross and diverge, they make the MACD oscillator to fluctuate above and below the zero line. A "signal line" is plotted on the top of the oscillator. It is a 9 day exponential moving average of MACD, which serves as a trigger for buy and sell signals. A sell signal is generated with the MACD going below the zero line.

MACD Histogram is an oscillator which measures the difference between the fast MACD line and the signal line. Just like MACD indicator, histogram also fluctuates above and below zero line. A positive histogram indicates that MACD is above its signal line, while MACD going below its signal line makes a negative histogram. A negative MACD Histograms generates sell signal.

To use MACD indicator and MACD Histogram in FinancialChart, add a FinancialChart control to your application and bind it to an appropriate data source or populate data in it through **Quote Collection**. The **ItemsSource** object enables data binding or populating data in FinancialChart. The MacdBase class exposes FastPeriod, SlowPeriod, and Smoothing period properties. Based on the values of these properties, data points for Macd indicator and Histogram are calculated and plotted on FinancialChart. The appearance of the series can be manipulated with MacdLineStyle and SignalLineStyle properties.

FinancialChart also enables you to fetch the calculated Macd values, Macd x values, Signal values, and Signal x values at run-time. This can help in creating alerts in application or maintaining logs while working with dynamic data.



The following example considers stock data for a company Box Inc. over a period of time and plots its MACD Indicator and MACD Histogram apart from the volume chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

PriceChart.xaml copyCode

```
<Window xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="MACDInd.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:MACDInd"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}">
<Grid>
    <c1:C1FinancialChart x:Name="financialChart"
                         ItemsSource="{Binding Data}"
                         BindingX="Date"
                         ChartType="Line"
                         ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
                         Margin="0,0,0,146"
                         Rendered="OnFinancialChartRendered">

        <c1:FinancialSeries Binding="High,Low,Open,Close" SeriesName="Box Inc." />
        <c1:C1FinancialChart.AxisX>
            <c1:Axis LabelAngle="45" MajorUnit="3"/>
        </c1:C1FinancialChart.AxisX>
    </c1:C1FinancialChart>
```

IndicatorChart.xaml copyCode

```
    <c1:C1FinancialChart x:Name="indicatorChart"
                         BindingX="Date"
                         Binding="High,Low,Close"
                         LegendPosition="Bottom"
                         ItemsSource="{Binding Data}"
                         Background="White"
                         ToolTipContent="{}{seriesName}&#x000A;Date: {Date}&#x000A;Y:
{y:n2}&#x000A;Volume: {Volume:n0}"
                         Margin="0,178,0,0">

        <c1:Macd x:Name="Macd" SeriesName="MACD,Signal">
            <c1:Macd.MacdLineStyle>
                <c1:ChartStyle Stroke="OrangeRed" />
            </c1:Macd.MacdLineStyle>
        </c1:Macd>

        <c1:MacdHistogram x:Name="MACDHistogram"
                          SeriesName= "MACD Histogram"
                          FastPeriod="12"
                          SlowPeriod="26"
                          SmoothingPeriod="9" />

        <c1:C1FinancialChart.AxisX>
        <c1:Axis LabelAngle="45" MajorUnit="3"/>
    </c1:C1FinancialChart.AxisX>
    </c1:C1FinancialChart>
</Grid>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "Indicator.Resources.box.json"
        'Replace Indicator by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "MACDInd.Resources.box.json";
        //Replace MACDInd by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
```

```
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for Macd.xaml
Partial Public Class Macd
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
        indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
    End Sub
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// Interaction logic for MainWindow.xaml
public partial class MainWindow : Window
{
    DataService dataService = DataService.GetService();
    public MainWindow()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }

    private void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
        indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
    }
}
```

Back to Top

Overlays

Technical overlays, like indicators, are a series of derived data points calculated by applying formulas to the historic and current price of financial instruments. They are used to forecast an asset's market direction. Unlike indicators, overlays are plotted with the original price or volume data, because their Y-axes scales are same.

The following sections discuss the technical overlays that FinanicalChart supports:

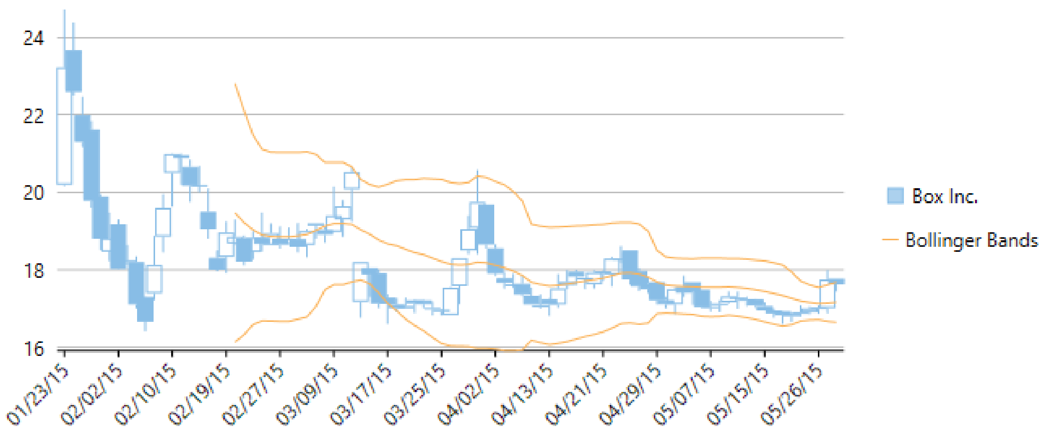
Bollinger Bands
    Learn about Bollinger Bands overlay.
Envelopes
    Learn about Envelopes overlay.

Bollinger Bands

Bollinger Bands overlay displays upper and lower limits of the price movements over the time. These are volatility bands based on standard deviation, and are placed above and below a moving average. Bollinger Bands are helpful in providing a relative definition of high and low price. Prices plotted near the upper band are considered high, while prices near the lower band are low. The width of the band depicts the measure of volatility. The price values touching the top of the Bollinger Band are supposed to be over extended above the intermediate trend, and the prices touching the bottom of the Bollinger band are supposed to be over extended below the intermediate trend.

To use Bollinger Bands overlay in FinancialChart, add a FinancialChart control to your application and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart. BollingerBands class exposes Multiplier property, to specify standard deviations for upper and lower bands, and IndicatorBase class provides the Period property, which takes integer value for calculating Simple Moving Average for the middle band. Based on the values of these properties, data points for Bollinger Bands overlay are calculated and plotted on FinancialChart.

FinancialChart also enables you to fetch the calculated Lower y values, Middle y values, Upper y values, and x values at run-time. This can help in creating alerts in application or maintaining logs while working with dynamic data.



The following example considers stock data for a company Box Inc. over a period of time and plots Bollinger Bands overlay on the same financial chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

Make sure that Build Action property of the json file is set to **Embedded Resource**.

XAML copyCode

```
    <Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:BollingerBOverlay"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="BollingerBOverlay.MainWindow"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
<c1:C1FinancialChart x:Name="financialChart"
                     ItemsSource="{Binding Data}"
                     Background="White"
                     ChartType="Candlestick"
                     BindingX="Date"
                     ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
                     Margin="6,37,4,71">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                SeriesName="Box Inc." />
            <c1:BollingerBands x:Name="bollinger"
                               Multiplier="2"
                               Period="20"
                               Binding="High,Low,Close"
                               SeriesName="Bollinger Bands" />

            <c1:C1FinancialChart.AxisX>
```

```
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "OverlayVB.Resources.box.json"
        'Replace OverlayVB by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
class DataService
{
    public List<Quote> GetData()
    {
        string path = "BollingerBOverlay.Resources.box.json";
        //Replace BollingerBOverlay by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic

- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for MWBollinger.xaml
Partial Public Class MWBollinger
    Inherits Window
    Private dataService As DataService = DataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
    [DataContract]
    public class Quote
    {
        [DataMember(Name = "date")]
        public string Date { get; set; }

        [DataMember(Name = "high")]
        public double High { get; set; }

        [DataMember(Name = "low")]
        public double Low { get; set; }

        [DataMember(Name = "open")]
        public double Open { get; set; }

        [DataMember(Name = "close")]
        public double Close { get; set; }

        [DataMember(Name = "volume")]
        public double Volume { get; set; }
    }
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        DataService dataService = DataService.GetService();
        public MainWindow()
        {
            InitializeComponent();
        }
        public List<Quote> Data
        {
            get
            {
                return dataService.GetData();
            }
        }
    }
}
```

Back to Top

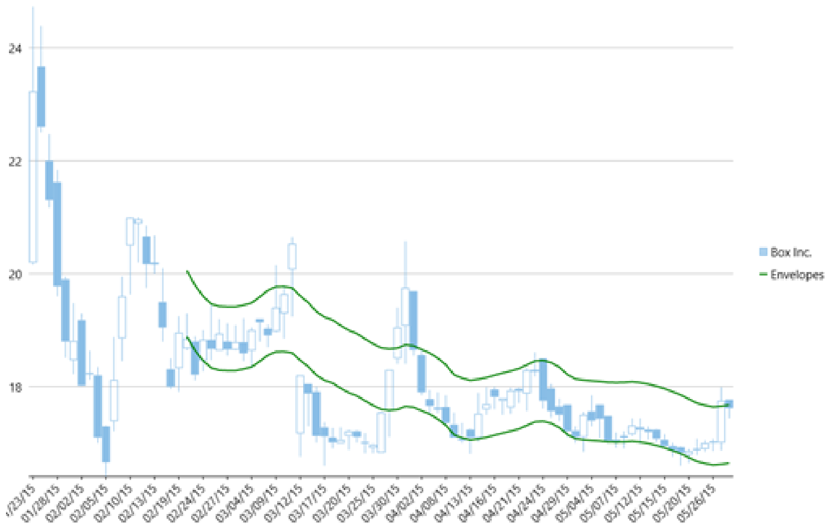
Envelopes

Envelopes overlay represent Moving Average Envelopes overlay series for the FinancialChart. These moving average envelopes are percentage based envelopes that are set above and below a standard moving average. The moving average could be simple or exponential moving average. During Bullish trends, breakthrough above the upper envelope signifies strength and that the uptrend will continue; during Bearish trends, breakthrough below the lower envelope signifies strength and that the downtrend will continue.

To use Envelopes overlay in FinancialChart, add a FinancialChart control to your application and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart.

Envelopes class exposes Size property, takes percentage values to render upper and lower envelopes, and Type property, to specify Simple or Exponential Moving average. IndicatorBase class provides the Period property, which takes integer value to specify base period for calculating simple or exponential moving average. Based on the values of these properties, data points for Envelopes overlays are calculated and plotted on FinancialChart.

FinancialChart also enables you to fetch the calculated Lower y values and Upper y values at run-time. This can help in creating alerts in application or maintaining logs while working with dynamic data.



The following example considers stock data for a company Box Inc. over a period of time and plots its Envelopes overlay on the same financial chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

XAML copyCode

```
    <Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:EnvelopeOverlay"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="EnvelopeOverlay.MainWindow"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             Background="White"
                             ChartType="Candlestick"
                             BindingX="Date"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
                             Margin="0,37,-1,71">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                SeriesName="Box Inc." />
          <c1:Envelopes Binding="High,Low,Close"
                        Period="20"
                        Size="0.03"
                        Type="Simple"
                        SeriesName="Envelopes">
            <c1:Envelopes.Style>
              <c1:ChartStyle Stroke="Green" StrokeThickness="2" />
            </c1:Envelopes.Style>
          </c1:Envelopes>
                <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "OverlayVB.Resources.box.json"
        'Replace OverlayVB by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "EnvelopeOverlay.Resources.box.json";
        //Replace EnvelopeOverlay by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for MWEnvelopes.xaml
Partial Public Class MWEnvelopes
    Inherits Window
    Private dataService As DataService = DataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
    public class Quote
    {
        [DataMember(Name = "date")]
        public string Date { get; set; }

        [DataMember(Name = "high")]
        public double High { get; set; }

        [DataMember(Name = "low")]
        public double Low { get; set; }

        [DataMember(Name = "open")]
        public double Open { get; set; }

        [DataMember(Name = "close")]
        public double Close { get; set; }

        [DataMember(Name = "volume")]
        public double Volume { get; set; }
    }

/// Interaction logic for MainWindow.xaml
public partial class MainWindow : Window
    {
        DataService dataService = DataService.GetService();
        public MainWindow()
        {
            InitializeComponent();
        }
        public List<Quote> Data
        {
            get
            {
                return dataService.GetData();
            }
        }
    }
```

Back to Top

Fibonacci Tools

Fibonacci tools enable the calculation and plotting of alert levels, in financial charts, useful in technical analysis. The mathematical relationships, ratios, between the numbers in Fibonacci series are used in technical analysis to help the traders of financial instruments in anticipating the changes in price and volume of these instruments.

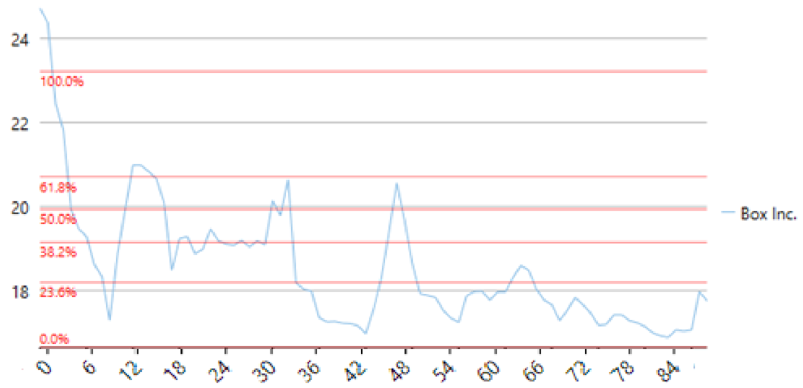
The following sections discuss the Fibonacci tools that are available in FinancialChart:

Fibonacci Retracements
    Learn how to implement Fibonacci Retracements in FinancialChart.
Fibonacci Arcs
    Learn how to implement Fibonacci Arcs in FinancialChart.
Fibonacci Fans
    Learn how to implement Fibonacci Fans in FinancialChart.
Fibonacci TimeZones
    Learn how to implement Fibonacci Time Zones in FinancialChart.

Fibonacci Retracements

Fibonacci Retracements are the technical analysis tools comprising of horizontal lines (or retracement levels) in a FinancialChart. These levels are used to identify the areas of support (stock prices won't go lower) or resistance (stock prices won't go higher) before the trend continues in original direction. These Fibonacci levels are created by first drawing a trend line between the high and low (two extremes), and then dividing the vertical distance by the key Fibonacci ratios of 23.6%, 38.2%, 50%, 61.8% and 100%. A vital Fibonacci retracement level is 61.8%, the maximum pullback zone, showing clear buy or sell signals.

To use Fibonacci Retracement tool in FinancialChart, add the control to your application, and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart. Fibonacci class exposes Uptrend and Levels properties. Based on the values you set for these properties, alert levels are plotted on FinancialChart.



The following example considers stock data for a company Box Inc. over a period of time and plots alert levels (retracements) on the same financial chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

XAML copyCode

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Fibonacci"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
    x:Class="Fibonacci.MainWindow"
    mc:Ignorable="d"
    Title="FibonacciRetracements"
    DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}">
<Grid>

    <c1:C1FinancialChart x:Name="financialChart"
                         ItemsSource="{Binding Data}"
                         BindingX="Date"
                         ChartType="Line"
                         ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
        <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>
        <c1:Fibonacci Binding="Close">
            <c1:Fibonacci.Style>
                <c1:ChartStyle Fill="Red"
                               Stroke="Red"
                               StrokeThickness="0.5"
                               FontSize="10"/>
            </c1:Fibonacci.Style>
        </c1:Fibonacci>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json

- System.Reflection
- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'Replace FibonacciVB by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Replace Fibonacci by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}

[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for MainWindow.xaml
Partial Public Class MainWindow
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}

public partial class MainWindow : Window
{
    DataService dataService = DataService.GetService();
    public MainWindow()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

Back to Top

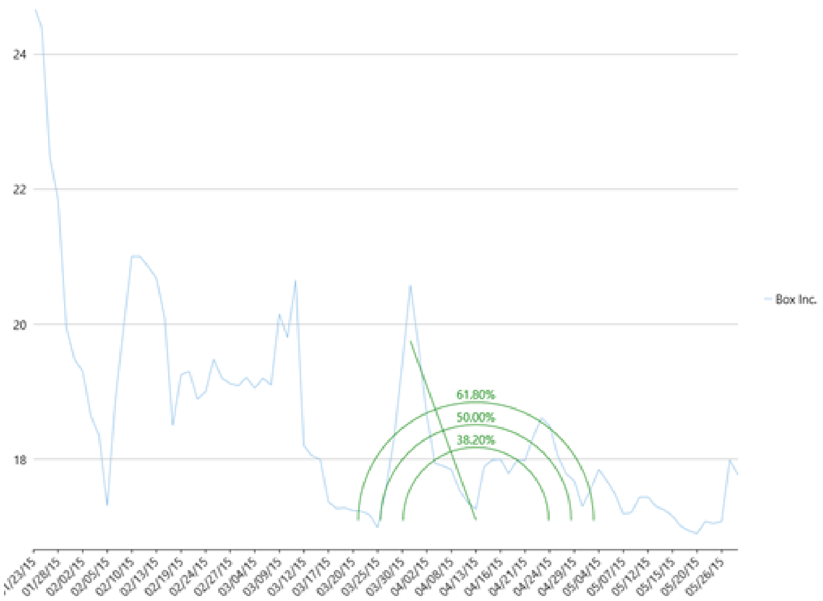
Fibonacci Arcs

Fibonacci Arcs are the technical analysis tools composed of a Base Line and three curved lines extending out from a trend line on FinancialChart. These arcs enable traders to predict resistance or reversal zones for counter trend bounce situations after decline. A small change in price over a short time period produces narrow arcs with a short Base Line; a big price move over a long time period produces wide arcs with a long Base Line.

To create Fibonacci Arcs a Base Line is drawn through two points, the high (peak) and the low (trough), in a given period. Arcs are drawn intersecting this base line at key Fibonacci levels 38.2%, 50%, and 61.8%. In contrast with Fibonacci Retracements, which are concerned only with change in price, Fibonacci Arcs consider time element too.

To use Fibonacci Arcs tool in FinancialChart, add the control to your application, and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart.

Fibonacci class exposes Uptrend property. Creating object of FibonacciArcs class enables Fibonacci arcs in a chart. Additionally, the FibonacciArcs class exposes StartX, EndX, StartY, and EndY properties. Based on the values of these properties, curves are plotted on FinancialChart.



The following example considers stock data for a company Box Inc. over a period of time and plots curves on the same financial chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

XAML copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindowArcs"
        mc:Ignorable="d"
        Title="FibonacciArcs"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        Height="300" Width="300">

    <Grid>
        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Line"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>
            <c1:FibonacciArcs x:Name="Arcs"
                              Binding="Close"
                              StartX="46"
                              StartY="19.75"
                              EndX="54"
                              EndY="17.1">
                <c1:FibonacciArcs.Style>
                    <c1:ChartStyle Stroke="Green" />
                </c1:FibonacciArcs.Style>
            </c1:FibonacciArcs>
            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
            </c1:C1FinancialChart>
    </Grid>
</Window>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'Replace FibonacciVB by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Replace Fibonacci by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}

[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }
    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for Arcs.xaml
Partial Public Class Arcs

    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}

public partial class MainWindowArcs : Window
{
    DataService dataService = DataService.GetService();
    public MainWindowArcs()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

Back to Top

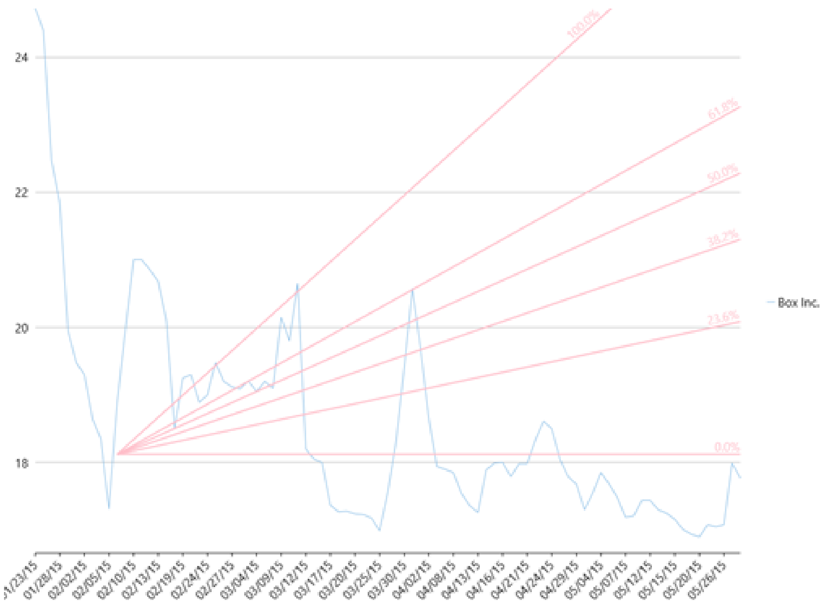
Fibonacci Fans

Fibonacci Fans are the rising and falling trend lines based on the Fibonacci retracement points. These technical indicators help in measuring the speed of a trend's movement, and also to construct support and resistance trend lines. The rising fan lines are used to predict support levels, while the falling fan lines can help predict resistance levels. In uptrend, if prices move below a Fibonacci Fan trend line, then they are expected to further fall until the next Fibonacci Fan trend line. In such cases, the Fibonacci Fan lines serve as support. Whereas, in a downtrend, if prices rise to a Fibonacci Fan trend line then that trend line is expected to serve as resistance.

To create Fibonacci Fans a trend line is first drawn through two points, the high and the low, in a given period and vertical distance between the two points is divided by the key Fibonacci ratios 38.2%, 50% and 61.8%. A point within the vertical distance is obtained as a result of each of these divisions. Three 'fan' lines are then created by drawing a line from the leftmost point to each of the three points representing a Fibonacci ratio. These are the trend lines that are based on Fibonacci retracement points.

To use Fibonacci Fans tool in FinancialChart, add the control to your application, and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart.

Fibonacci class exposes Uptrend property. Creating object of FibonacciFans class enables Fibonacci fans in a chart. Additionally, the **FibonacciFans** class exposes StartX, EndX, StartY, and EndY properties. Based on the values of these properties, Fibonacci Fan lines are plotted on FinancialChart.



The following example considers stock data for a company Box Inc. over a period of time and plots fan lines on the same financial chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

XAML copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindowFans"
        mc:Ignorable="d"
        Title="FibonacciFans"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        Height="300" Width="300">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
```

```
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Line"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>
            <c1:FibonacciFans x:Name="fans"
                            Binding="Close"
                            StartX="10"
                            StartY="18.12"
                            EndX="32"
                            EndY="20.53">

                <c1:FibonacciFans.Style>
                    <c1:ChartStyle Stroke="Pink" />
                </c1:FibonacciFans.Style>
            </c1:FibonacciFans>

            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'Replace FibonacciVB by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Replace Fibonacci by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
```

```
            _ds = new DataService();
        return _ds;
    }
}

[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for Fans.xaml
Partial Public Class Fans
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }
    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}

public partial class MainWindowFans : Window
{
    DataService dataService = DataService.GetService();
    public MainWindowFans()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

Back to Top

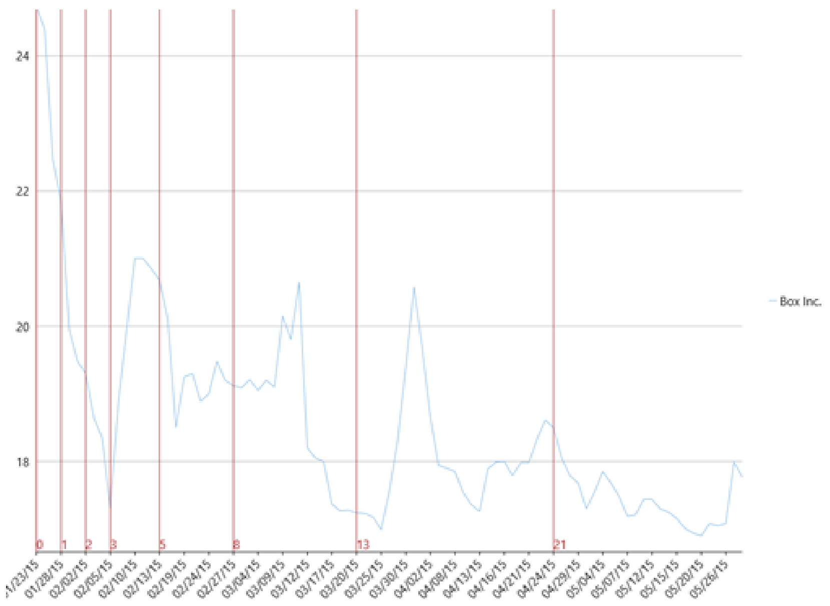
Fibonacci Time Zones

Fibonacci Time Zones are the technical indicators used by traders to forecast the periods when the price of a financial instrument will show significant move. The Time Zones are series of vertical lines that correspond to a sequence of numbers, called Fibonacci numbers (1, 2, 3, 5, 8, 13, 21, 34, and so on.).

A starting position for the time zones (following a major price move) on financial chart is decided by a trader, and a vertical line is placed on every subsequent day that corresponds to the position in the Fibonacci number sequence. The Fibonacci Time Zones enable technical traders to anticipate future price changes near these vertical time zone lines, and take buy/ sell decisions. Ideally, traders are advised to ignore initial 7 to 8 time zones, as the potential price reversal points are found ahead of 21, 34, 55, 89, and 144 days, which correspond to the 8th , 9th, 10th, 11th, and 12th time zones.

To use Fibonacci Time Zones in FinancialChart, add the control to your application and bind it to an appropriate data source or populate data in it through **Quote Collection**. ItemsSource object enables data binding or populating data in FinancialChart.

Fibonacci class exposes Uptrend property. Creating object of FibonacciTimeZones class enables Fibonacci time zones in a chart. Additionally, the **FibonacciTimeZones** class exposes StartX and EndX properties. Based on the values of these properties, time zone lines are plotted on FinancialChart.



The following example considers stock data for a company Box Inc. over a period of time and plots time zone lines on the same financial chart, as shown in the image above. The example uses data from a json file, and DataService.cs class is created to access this json file.

⚠ Make sure that Build Action property of the json file is set to **Embedded Resource**.

XAML copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindowTZ"
        mc:Ignorable="d"
        Title="FibonacciTimeZones"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        Height="300" Width="300">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Line"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>
            <c1:FibonacciTimeZones x:Name="timeZones" Binding="Close" StartX="0" EndX="3">
                <c1:FibonacciTimeZones.Style>
                    <c1:ChartStyle Stroke="Brown" />
                </c1:FibonacciTimeZones.Style>
            </c1:FibonacciTimeZones>
            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'Replace FibonacciVB by your application name
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Replace Fibonacci by your application name
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}

[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
```

Json Data

```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
    { "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
    { "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
    { "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
    { "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
    { "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
    { "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
    { "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
    { "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
    { "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
    { "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
    { "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
    { "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
    { "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
    { "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
    { "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
    { "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
    { "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
    { "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
    { "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
    { "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
    { "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
    { "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
    { "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
    { "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
    { "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
    { "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
    { "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
    { "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
    { "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
    { "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
    { "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
    { "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
    { "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
    { "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
    { "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
    { "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
    { "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
    { "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
    { "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
    { "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
    { "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
    { "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
    { "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
    { "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
    { "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
    { "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
    { "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
    { "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
    { "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
    { "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
    { "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
    { "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
    { "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
            ]
```

Make sure to add the following references in code view:

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String
    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Interaction logic for TimeZones.xaml
Partial Public Class TimeZones

    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}

public partial class MainWindowTZ : Window
{
    DataService dataService = DataService.GetService();
    public MainWindowTZ()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

Back to Top

Interaction

In **FinancialChart for WPF**, there is a set of interactive built-in tools that help you customize and further develop applications. Features like Range Selector allows end users to adjust the FinancialChart's visible range of data at runtime.

To know more about Range Selector, click the following link:

- Range Selector

Range Selector

FinancialChart's RangeSelector lets a user select a specific range of data to be displayed on the chart. A user can easily bind the RangeSelector with various types of financial charts. It is mostly used by finance industry to perform stock analysis on different data ranges.

The RangeSelector has a left thumb (for minimum value) and right thumb (for maximum value) that lets you scroll through particular time periods on the chart. Users can change the minimum and maximum values of the RangeSelector, and adjust the visible range of data on the chart by dragging these thumbs to left or right. On dragging the thumb towards left on the range bar, you reduce its value, and dragging it towards the right increases its value on the range bar.

The following code snippet shows how you can use RangeSelector to create your applications.

- **DataService.cs**

```
public class DataService
{
    List<Company> _companies = new List<Company>();
    Dictionary<string, List<Quote>> _cache = new Dictionary<string, List<Quote>>();

    private DataService()
    {
        _companies.Add(new Company() { Symbol = "box", Name = "Box Inc" });
        _companies.Add(new Company() { Symbol = "fb", Name = "Facebook" });
    }

    public List<Company> GetCompanies()
    {
        return _companies;
    }

    public List<Quote> GetSymbolData(string symbol)
    {
        if (!_cache.Keys.Contains(symbol))
        {
            string path = string.Format("FinancialChartExplorer.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            _cache.Add(symbol, data.ToList());
        }

        return _cache[symbol];
    }

    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

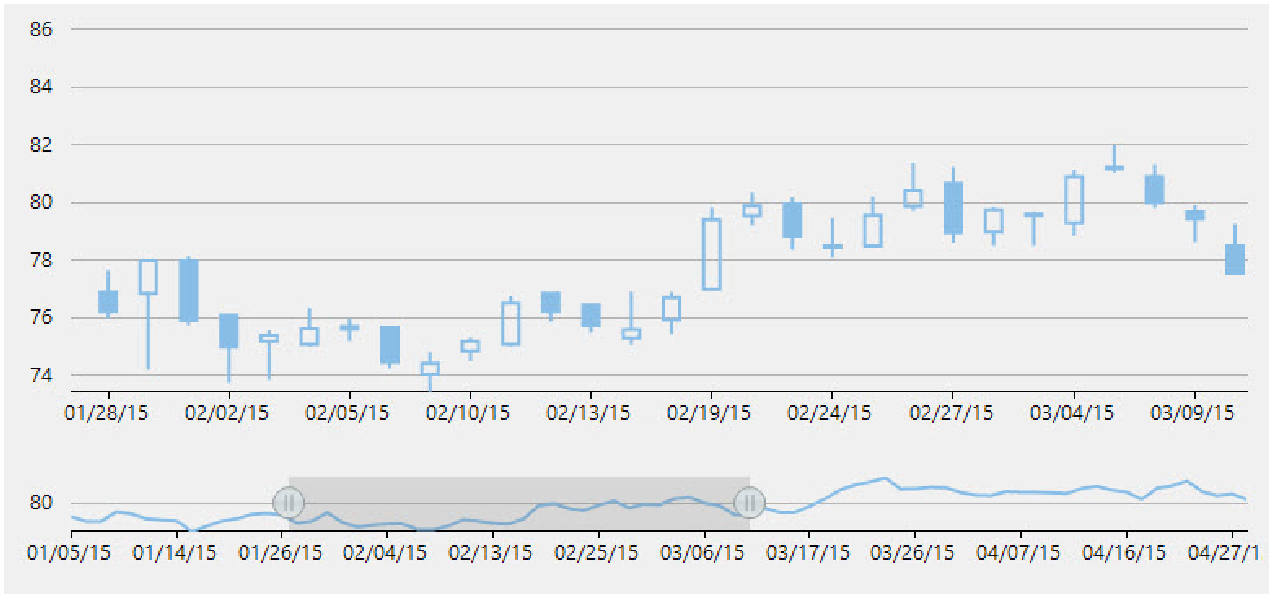
XAML

```
<c1:C1FinancialChart BindingX="date"
                     Binding="high,low,open,close,volume"
                     ChartType="Candlestick"
                     ItemsSource="{Binding Data}">
    <c1:FinancialSeries />
    <c1:C1FinancialChart.AxisX>
        <c1:Axis Min="{Binding Source={x:Reference Name=rangeSelector},
            Path=LowerValue}"
                 Max="{Binding Source={x:Reference Name=rangeSelector},
            Path=UpperValue}" />
    </c1:C1FinancialChart.AxisX>
</c1:C1FinancialChart>
```

Code

C# copyCode

```
public partial class RangeSelector : UserControl
{
    DataService dataService = DataService.GetService();

    public RangeSelector()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("fb");
        }
    }
}
```


86
84
82
80
78
76
74
01/28/15
02/02/15
02/05/15
02/10/15
02/13/15
02/19/15
02/24/15
02/27/15
03/04/15
03/09/15
80
01/05/15
01/14/15
01/26/15
02/04/15
02/13/15
02/25/15
03/06/15
03/17/15
03/26/15
04/07/15
04/16/15
04/27/1