



Essential Studio 2013 Volume 4 - v.11.4.0.26

## Essential Chart for Windows Forms



# Contents

<b>1</b>	<b>Overview</b>	<b>11</b>
1.1	Introduction to Essential Chart .....	11
1.2	Prerequisites and Compatibility .....	14
1.3	Documentation .....	14
<b>2</b>	<b>Installation and Deployment</b>	<b>16</b>
2.1	Installation.....	16
2.2	Sample and Location.....	16
2.3	Deployment Requirements.....	19
<b>3</b>	<b>Getting Started</b>	<b>20</b>
3.1	Control Structure .....	20
3.2	Creating a simple Chart.....	22
<b>4</b>	<b>Concepts and Features</b>	<b>26</b>
4.1	Chart Wizard.....	26
4.1.1	Chart Type.....	28
4.1.2	Series.....	29
4.1.3	Appearance .....	33
4.1.4	Axes.....	35
4.1.5	Points.....	37
4.1.6	Toolbar .....	38
4.1.7	Legend.....	41
4.2	Chart Data .....	43
4.2.1	Binding a DataSet to the Chart.....	44
4.2.2	Implementing Custom Data Binding Interfaces.....	47
4.2.3	Chart Data Binding with IEnumerables .....	50
4.2.4	Data Binding in Chart Through Chart Wizard.....	54
4.3	Improving Performance .....	67
4.4	Chart Types .....	70
4.4.1	Line Charts .....	72
4.4.1.1	Line Chart.....	72
4.4.1.2	Spline Chart.....	74

4.4.1.3 Rotated Spline Chart .....	76
4.4.1.4 Step Line Chart.....	78
4.4.2 Bar Charts .....	80
4.4.2.1 Bar Chart .....	80
4.4.2.2 Stacking Bar Chart .....	83
4.4.2.3 StackedBar100 Chart .....	85
4.4.2.4 Gantt Chart.....	87
4.4.2.5 Histogram Chart .....	88
4.4.2.6 Tornado Chart .....	91
4.4.3 Column Charts.....	93
4.4.3.1 Column Chart .....	93
4.4.3.2 Column Range Chart.....	95
4.4.3.3 Stacking Column Chart .....	97
4.4.3.4 Stacked Column100 Chart .....	99
4.4.4 Area Charts .....	101
4.4.4.1 Area Chart .....	102
4.4.4.2 Spline Area Chart .....	103
4.4.4.3 Stacking Area Chart .....	106
4.4.4.4 StackedArea100 Chart .....	108
4.4.4.5 Step Area Chart.....	110
4.4.4.6 Range Area Chart.....	112
4.4.5 Accumulation Charts .....	114
4.4.5.1 Funnel Chart.....	114
4.4.5.2 Pyramid Chart.....	118
4.4.6 XY Charts (Bubble and Scatter) .....	121
4.4.6.1 Scatter Chart .....	122
4.4.6.2 Bubble Chart.....	124
4.4.7 Financial Charts.....	126
4.4.7.1 Candle Chart .....	127
4.4.7.2 Hi Lo Chart .....	129
4.4.7.3 Hi Lo Open Close Chart .....	130
4.4.7.4 Kagi Chart.....	132
4.4.7.5 Point and Figure Chart .....	135
4.4.7.6 Renko Chart .....	137
4.4.7.7 Three Line Break Chart.....	140

4.4.7.8	Box And Whisker Chart .....	142
4.4.8	Pie Chart.....	145
4.4.8.1	Doughnut Chart .....	147
4.4.9	Polar And Radar Chart .....	148
4.4.9.1	Polar Chart .....	149
4.4.9.2	Radar Chart .....	151
4.4.10	Combination Chart.....	153
4.4.11	Heat Map Charts .....	155
4.4.12	Stacking Charts .....	158
4.4.13	Step Charts.....	158
4.4.14	Sparkline.....	159
4.5	Chart Series.....	168
4.5.1	Series Customization.....	171
4.5.1.1	AngleOffset.....	177
4.5.1.2	Border	179
4.5.1.3	BubbleType .....	182
4.5.1.4	ColumnDrawMode.....	184
4.5.1.5	ColumnWidthMode .....	187
4.5.1.6	ColumnFixedWidth .....	191
4.5.1.7	ColumnType .....	193
4.5.1.8	ColorsMode .....	195
4.5.1.9	DarkLightPower .....	196
4.5.1.10	DisplayShadow .....	198
4.5.1.11	DisplayText.....	200
4.5.1.12	DoughnutCoeficient .....	203
4.5.1.13	DrawColumnSeparatingLines.....	204
4.5.1.14	DrawErrorBars .....	206
4.5.1.15	DrawHistogramNormalDistribution .....	210
4.5.1.16	DrawSeriesNameInDepth.....	211
4.5.1.17	DropSeriesPoints.....	213
4.5.1.18	ElementBorders .....	214
4.5.1.19	EnablePhongStyle .....	216
4.5.1.20	EnableAreaToolTip.....	217
4.5.1.21	ErrorBarsSymbolShape .....	218
4.5.1.22	ExplodedAll.....	220

4.5.1.23	ExplodedIndex.....	221
4.5.1.24	ExplosionOffset .....	222
4.5.1.25	FancyToolTip.....	224
4.5.1.26	FigureBase .....	225
4.5.1.27	FillMode .....	230
4.5.1.28	FunnelMode.....	232
4.5.1.29	Font .....	235
4.5.1.30	GanttDrawMode .....	237
4.5.1.31	GapRatio .....	239
4.5.1.32	Gradient.....	241
4.5.1.33	HeightBox .....	243
4.5.1.34	HeightByAreaDepth.....	245
4.5.1.35	HeightCoeficient .....	247
4.5.1.36	HighlightInterior .....	248
4.5.1.37	HitTestRadius .....	251
4.5.1.38	ImageIndex.....	253
4.5.1.39	Images.....	255
4.5.1.40	InSideRadius .....	257
4.5.1.41	Interior .....	259
4.5.1.42	LabelPlacement.....	262
4.5.1.43	LabelStyle.....	265
4.5.1.44	LegendItem.....	267
4.5.1.45	LightAngle.....	268
4.5.1.46	LightColor .....	270
4.5.1.47	Name .....	271
4.5.1.48	NumberOfHistogramIntervals .....	273
4.5.1.49	OpenCloseDrawMode .....	274
4.5.1.50	OptimizePiePointPositions .....	278
4.5.1.51	PhongAlpha .....	281
4.5.1.52	PieType .....	283
4.5.1.53	PieWithSameRadius .....	285
4.5.1.54	PointsToolTipFormat .....	286
4.5.1.55	PointWidth .....	286
4.5.1.56	PriceDownColor.....	288
4.5.1.57	PriceUpColor .....	290

4.5.1.58	PyramidMode .....	292
4.5.1.59	Radar Type.....	293
4.5.1.60	RadarStyle.....	295
4.5.1.61	RelatedPoints .....	297
4.5.1.62	ReversalAmount.....	300
4.5.1.63	Rotate .....	302
4.5.1.64	ScatterConnectType.....	304
4.5.1.65	ScatterSplineTension .....	306
4.5.1.66	SeriesToolTipFormat.....	308
4.5.1.67	ShadingMode .....	310
4.5.1.68	ShadowInterior .....	311
4.5.1.69	ShadowOffset.....	314
4.5.1.70	ShowDataBindLabels .....	316
4.5.1.71	ShowHistogramDataPoints .....	319
4.5.1.72	ShowTicks .....	321
4.5.1.73	SmartLabels .....	323
4.5.1.74	Spacing.....	325
4.5.1.75	SpacingBetweenSeries .....	327
4.5.1.76	SpacingBetweenPoints .....	329
4.5.1.77	Stacking Group.....	330
4.5.1.78	StepItem.Inverted .....	332
4.5.1.79	Summary .....	334
4.5.1.80	Symbol.....	336
4.5.1.81	Text (Series) .....	340
4.5.1.82	Text (Style) .....	342
4.5.1.83	TextColor .....	344
4.5.1.84	TextFormat .....	346
4.5.1.85	TextOffset.....	348
4.5.1.86	TextOrientation .....	349
4.5.1.87	ToolTip.....	352
4.5.1.88	ToolTipFormat .....	355
4.5.1.89	Visible .....	358
4.5.1.90	VisibleAllPies .....	360
4.5.1.91	XType .....	362
4.5.1.92	YType .....	363

4.5.1.93	ZOrder .....	365
4.5.2	Data Point Labels, Tooltips and Symbols.....	368
4.5.3	Custom Points .....	369
4.5.3.1	Custom Point in Multiple Axes.....	374
4.5.4	Empty Points .....	375
4.6	Chart Axes.....	379
4.6.1	Indexed X Values .....	380
4.6.2	Inverted Axis.....	382
4.6.3	Opposed Axis .....	383
4.6.4	Multiple Axes .....	384
4.6.5	Axis Value Type.....	389
4.6.6	Axis Range and Intervals .....	390
4.6.7	Axis Dimensions .....	394
4.6.8	Axis Labels .....	397
4.6.8.1	Axis Label Text Formatting, Appearance and Positioning .....	397
4.6.8.2	Customizing Label Text.....	399
4.6.8.3	Intersecting Labels .....	405
4.6.8.4	Grouping Labels .....	406
4.6.8.5	Tooltip Support for ChartAxisLabels.....	408
4.6.9	Axis Title .....	412
4.6.10	Axis Ticks .....	415
4.6.11	3-D Related .....	418
4.6.12	Chart Grid Lines .....	422
4.6.13	Chart StripLines.....	424
4.6.14	Chart Breaks.....	428
4.6.15	Axis Crossing Support.....	431
4.6.16	Axis Label Placement.....	432
4.7	Chart Area .....	437
4.8	Chart Legend and Legend Items.....	438
4.8.1	ChartLegend.....	439
4.8.2	ChartLegendItem.....	443
4.8.3	Customizing LegendItem Image.....	450
4.9	Runtime Features .....	458
4.9.1	Zooming and Scrolling.....	458
4.9.2	Toolbars.....	464

4.9.2.1 Toolbar Properties .....	467
4.9.2.2 Appearance .....	469
4.9.3 Context Menu .....	472
4.9.4 Interactive Features.....	474
4.9.4.1 Drawing Interactive Cursor Separately – Either Horizontally or Vertically or Both .....	478
4.9.4.2 ChartInteractiveCursor Support for Chart Area.....	484
4.9.5 ToolTips .....	486
4.10 Chart Appearance .....	491
4.10.1 Background Colors .....	491
4.10.2 Background Image .....	494
4.10.3 Border and Margins .....	497
4.10.4 Foreground Settings.....	502
4.10.5 Multiple Chart Titles.....	504
4.10.6 Custom Drawing .....	509
4.10.7 Watermark Support .....	511
4.10.8 Interlaced Grid Background.....	513
4.10.9 Minor Grid Lines .....	514
4.10.10 Chart Skins .....	516
4.11 Realtime.....	525
4.12 Statistical Formulas .....	526
4.12.1 Basic Statistical Formulas .....	526
4.12.1.1 Anova Test .....	527
4.12.1.2 Correlation .....	531
4.12.1.3 Covariance .....	533
4.12.1.4 F-Test .....	535
4.12.1.5 Mean.....	536
4.12.1.6 Median.....	537
4.12.1.7 Standard Deviation.....	538
4.12.1.8 T-Tests .....	539
4.12.1.8.1 TTest with Equal Variances.....	540
4.12.1.8.2 T-Test with UnEqual Variance .....	542
4.12.1.8.3 TTest Paired.....	544
4.12.1.9 Variance .....	546
4.12.1.10 Z-Test .....	547

4.12.2	Utility Functions .....	549
4.12.2.1	Beta Function .....	550
4.12.2.2	Beta Cumulative Distribution .....	551
4.12.2.3	Binomial Coefficient.....	553
4.12.2.4	Inverse Beta Cumulative Distribution .....	554
4.12.2.5	Error Function.....	555
4.12.2.6	Factorial.....	556
4.12.2.7	F Cumulative Distribution .....	557
4.12.2.8	Gamma Function .....	558
4.12.2.9	Gamma Cumulative Distribution.....	560
4.12.2.10	Inverse Error Function .....	561
4.12.2.11	Inverse F Cumulative Distribution .....	561
4.12.2.12	Inverse Normal Distribution.....	562
4.12.2.13	Normal Distribution.....	563
4.12.2.14	Normal Distribution Density.....	565
4.12.2.15	Inverse T Cumulative Distribution .....	567
4.12.2.16	TCumulative Distribution .....	567
4.13	Importing.....	569
4.13.1	Importing a CSV file.....	569
4.13.2	Import Data from XML to a Chart .....	570
4.13.3	Import Data from Excel to Chart.....	570
4.14	Exporting.....	571
4.14.1	Exporting as an Image.....	571
4.14.2	Exporting to Word Doc .....	574
4.14.3	Exporting to Grid.....	577
4.14.4	Exporting to Excel.....	581
4.14.5	Exporting to PDF .....	584
4.15	Design time Features .....	586
4.15.1	Chart Templates .....	586
4.15.2	Tasks Window .....	590
4.16	Printing and Print Preview .....	594
4.17	Data Manipulation.....	597
4.18	Hit Testing.....	600
4.18.1	Chart Coordinates by Point .....	600
4.18.2	LegendItem By Point .....	601

4.18.3	Chart Area Bounds .....	603
4.19	Chart control events .....	605
4.19.1	Chart Region Events .....	606
[C#]	608	
4.19.2	VisibleRangeChanged Event .....	609
4.19.3	ChartFormatAxisLabel Event .....	610
4.19.4	PrepareStyle Event.....	610
4.19.5	SeriesInCompatible Event.....	611
4.19.6	LayoutCompleted Event .....	612
4.19.7	ChartAreaPaint Event.....	612
4.19.8	ChartLegendFilterItems Event.....	613
4.19.9	PreChartAreaPaint Event .....	613
4.20	Localization.....	613
<b>5</b>	<b>Frequently Asked Questions</b>	<b>617</b>
5.1	How to add custom TrendLine in Chart.....	617
5.2	How to customize the data points for Chart Series .....	618
5.3	How to display custom tooltip over Histogram Chart columns .....	619
5.4	How to drag the Chart Series Points at run time .....	620
5.5	How to draw the Y-axis at the center of the X-axis .....	622
5.6	How to export the Chart Points into a CSV file.....	622
5.7	How to filter particular set of data points in the Chart Series .....	624
5.8	How to hide the Chart ZoomButton .....	626
5.9	How to show or hide the tabs in the series style dialog in a Chart.....	627
5.10	How to display the Chart Area alone .....	628
5.11	How to get back to the gradient appearance of the Chart Series .....	628
5.12	How to implement DrillDown charts .....	630
5.13	How to provide input data of DateTime type .....	630
5.14	How to set the Color Palette for a Chart.....	631
5.15	How to specify the position for a Floating Legend .....	632
5.16	How to find value, maximum value and minimum value of the data points .....	633
5.17	How to Print a Chart in Multiple Pages.....	635

## 1 Overview

---

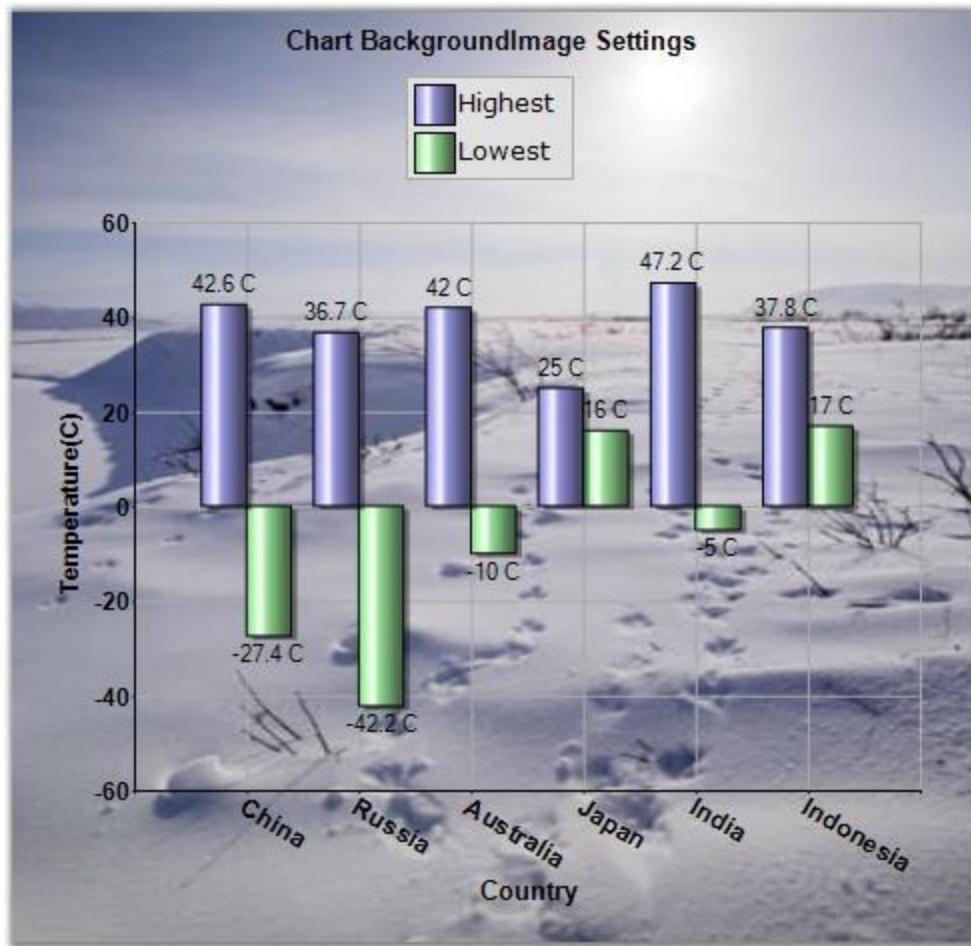
This section covers information on Essential Chart control, its key features, prerequisites to use the control, its compatibility with various OS and browsers and finally the documentation details complimentary with the product. It comprises the following sub sections:

### 1.1 Introduction to Essential Chart

Essential Chart is an easily configurable, presentation quality business chart control. A unique point based Styles Architecture, easily extendable Data Model, easily extendable rendering model, native Date Handling and a unique Shared Data Model are the features that distinguish Essential chart. This control has been created using C#, which means that it is a fully-managed .NET component, and has been specifically designed for use with Microsoft's Visual Studio .NET.

Essential Chart for Windows Forms is a perfect solution for developers looking to add advanced, feature rich, visually appealing charts to their Windows Forms applications. The product comes with numerous samples as well as an extensive documentation to guide you every step of the way.

Chart is used as a means to show the graphical representation of two values. For Example, a Line Chart can be used in health Statistic reports, in which it can show the rates over time or for a series of values, such as age-specific death rates. Logarithmic charts can be used in Share price charts where we plot between price and time. In logarithmic chart, we can identify the 'Proportional change in price' with respect to the 'Change in time'. We use 'Proportional change in price' to observe market sentiment. Market participants move share prices either up or down proportionally depending on how bullish or bearish they feel.



*Figure 1: Chart BackgroundImage Settings*

## **Key Features**

Some of the key features of Essential Chart are listed below.

- The control provides complete support for customization of the Chart control through Chart Wizard at design time and also at run time. Chart Wizard comes with new Office look and feel.
- Chart Data Model is an innovative data object model which makes it easy to populate the chart with any kind of data source.
- Essential Chart features built in support for dates. The data type of any series that is plotted on the chart can be set to DateTime.
- Essential Chart offers automatic interval calculation capabilities for any range of numbers or dates. This calculation can be overridden by explicit allocation of ranges and intervals

to be used and also with settings that control how 'nice' numbers are calculated for display.

- Essential Chart offers extensive customization possibilities of the legend. The position of the legend on the chart area as well as its representation aspects can all be completely customized. Essential Chart also features modification of legend items using events. It also supports custom legend items that are not tied to series of data.
- Statistical formula Mean, Standard Deviation, Variance, Distributions, t-test, f-test and Z-test etc., support.
- Exporting Chart to PDF, to Excel and to Doc etc., are available for the chart control. Importing is also supported.
- Users can create custom palettes for their Charts using CustomPalette property. Also, create non-gradient palettes for the Charts using this custom palettes feature.
- Multiple Chart Titles and Multiple Legends can be provided with abilities to format the Title text.
- Chart Breaks are introduced in this version. Breaks are very useful when you use series points with large difference.

## User Guide Structure

The product comes with numerous samples as well as an extensive documentation to guide you. This User Guide provides detailed information on the features and functionalities of the Tools controls. It is organized into the following sections:

- **Overview**-This section gives a brief introduction to our product and its key features.
- **Installation and Deployment**-This section elaborates on the install location of the samples, license etc.
- **What's New**-This section lists the new features implemented for every release.
- **Getting Started**-This section guides you on getting started with Windows application, controls etc.
- **Concepts and Features**-The features of Essential Chart are illustrated with use case scenarios, code examples and screen shots under this section.

## Document Conventions

The conventions listed below will help you to quickly identify the important sections of information, while using the content:

Convention	ICON	Description
Note	Note:	Represents important information
Example	Example	Represents an example
Tip		Represents useful hints that will help you in using the controls/features

Important Note		Represents additional information on the topic
----------------	-----------------------------------------------------------------------------------	------------------------------------------------

## 1.2 Prerequisites and Compatibility

This section covers the requirements mandatory for using Essential Chart control. It also lists operating systems and browsers compatible with the product.

### Prerequisites

The prerequisites details are listed in the following table.

Development Environments	<ul style="list-style-type: none"><li>• Visual Studio 2013</li><li>• Visual Studio 2012</li><li>• Visual Studio 2010 (Ultimate and Express)</li><li>• Visual Studio 2008 (Team, Professional, Standard, and Express)</li><li>• Visual Studio 2005 (Team, Professional, Standard, and Express)</li><li>• Borland Delphi for .NET</li><li>• SharpCode</li></ul>
.NET Framework versions	<ul style="list-style-type: none"><li>• .NET 4.5 RTM</li><li>• .NET 4.0</li><li>• .NET 3.5</li><li>• .NET 2.0</li></ul>

### Compatibility

The compatibility details are listed in the following table.

Operating Systems	<ul style="list-style-type: none"><li>• Windows 8.1 (32 bit and 64 bit)</li><li>• Windows Server 2008 (32 bit and 64 bit)</li><li>• Windows 7 (32 bit and 64 bit)</li><li>• Windows Vista (32 bit and 64 bit)</li><li>• Windows XP</li><li>• Windows 2003</li></ul>
-------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 1.3 Documentation

Syncfusion provides the following documentation segments to provide all necessary information for using Essential Chart control in Windows application in an efficient manner.

Type of documentation	Location
Readme	[drive:]\\Program Files\\Syncfusion\\Essential Studio\\x.x.x.x\\Infrastructure\\Data\\Release Notes\\readme.htm
Release Notes	[drive:]\\Program Files\\Syncfusion\\Essential Studio\\x.x.x.x\\Infrastructure\\Data\\Release Notes\\Release Notes.htm
User Guide (this document)	<b>Online</b> <a href="http://help.syncfusion.com/resources">http://help.syncfusion.com/resources</a> (Navigate to the Chart for Windows Forms User Guide.)  <b>Note:</b> Click Download as PDF to access a PDF version.  <b>Installed Documentation</b> Dashboard -> Documentation -> Installed Documentation.
Class Reference	<b>Online</b> <a href="http://help.syncfusion.com/resources">http://help.syncfusion.com/resources</a> (Navigate to the Windows Forms User Guide. Select <i>Chart</i> in the second text box, and then click the Class Reference link found in the upper right section of the page.)  <b>Installed Documentation</b> Dashboard -> Documentation -> Installed Documentation.

## **2 Installation and Deployment**

---

This section covers information on the install location, samples, licensing, patches update and updation of the recent version of Essential Studio. It comprises the following sub-sections:

### **2.1 Installation**

For step-by-step installation procedure for the installation of Essential Studio, refer to the **Installation** topic under **Installation and Deployment** in the **Common UG**.

#### **See Also**

For licensing, patches and information on adding or removing selective components, refer the following topics in Common UG under Installation and Deployment.

- Licensing
- Patches
- Add / Remove Components

### **2.2 Sample and Location**

This section covers the location of the installed samples and describes the procedure to run the samples through the sample browser. It also lists the location of source code.

#### **Sample Installation Location**

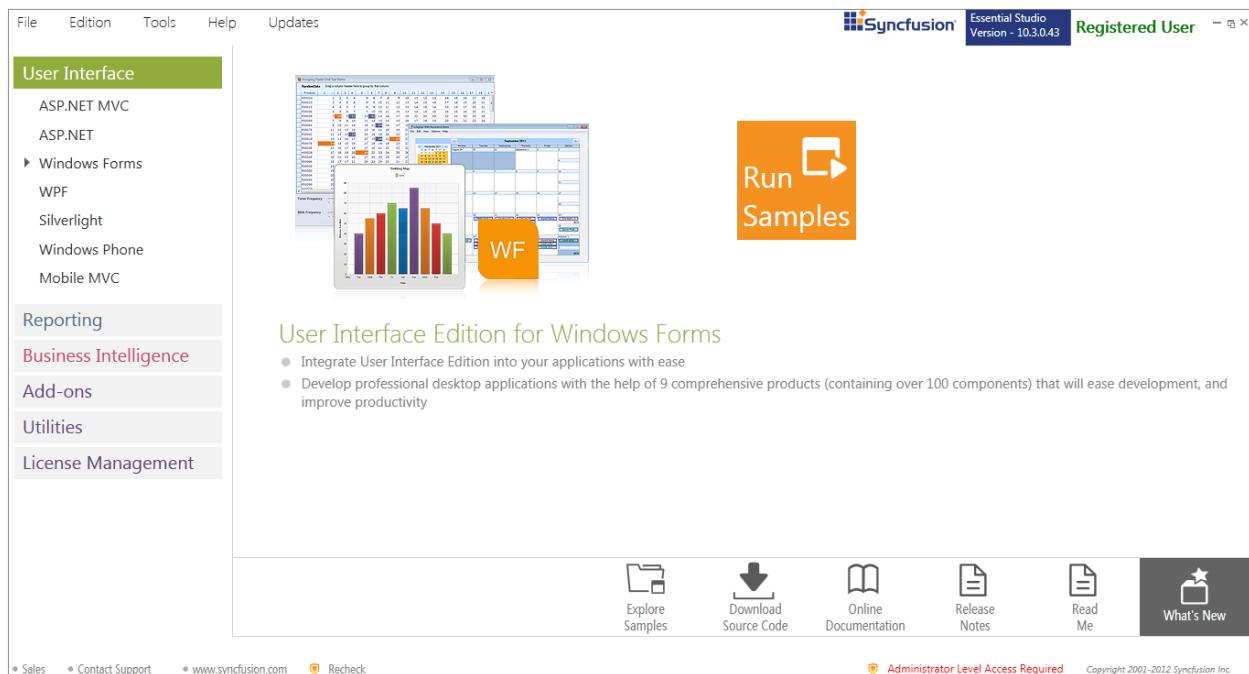
The Chart Windows Forms samples are installed in the following location:

[...\\My Documents\\Syncfusion\\EssentialStudio\\Version Number\\Windows\\Chart.Windows\\Samples\\2.0](http://...\\My Documents\\Syncfusion\\EssentialStudio\\Version Number\\Windows\\Chart.Windows\\Samples\\2.0)

#### **Viewing Samples**

To view the samples, follow the steps below:

1. Click Start-->All Programs-->Syncfusion-->Essential Studio <version number> -->Dashboard.



*Figure 2: Syncfusion Essential Studio Dashboard*

2. In the Dashboard window, click **Run Samples** for Windows Forms under UI Edition. The UI Windows Forms Sample Browser window is displayed.



**Note:** You can view the samples in any of the following three ways:

- **Run Samples**-Click to view the locally installed samples
- **Online Samples**-Click to view online samples
- **Explore Samples**-Explore BI Web samples on disk

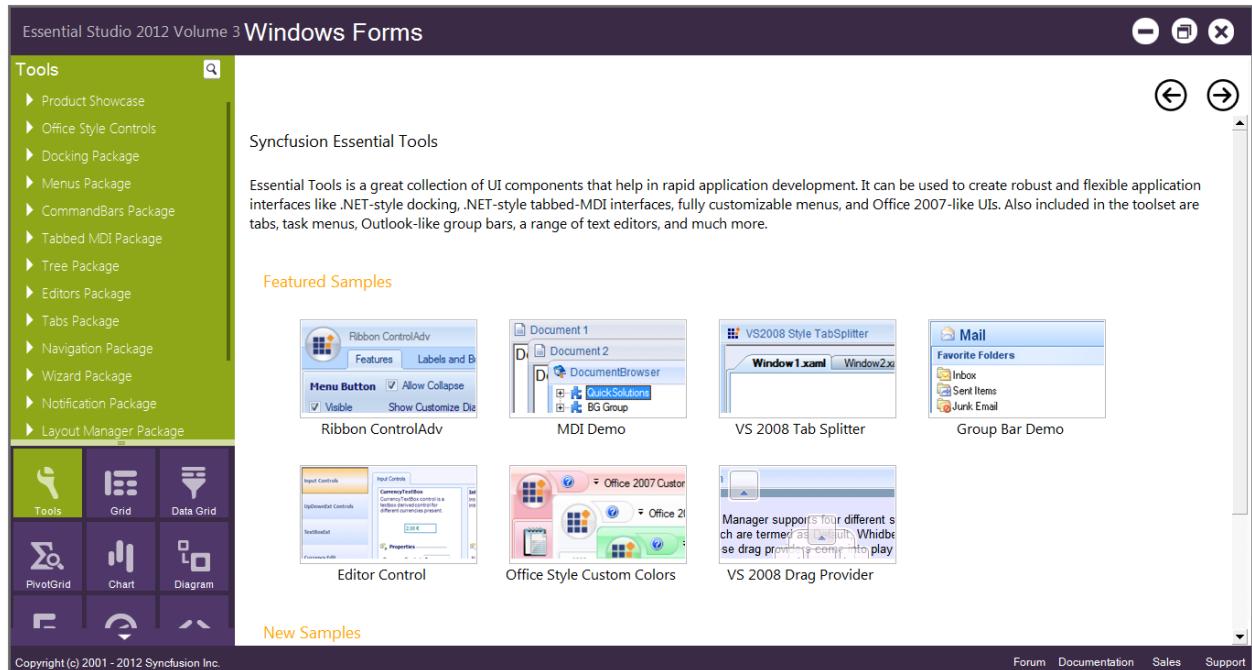


Figure 3: User Interface Edition Windows Forms Sample Browser

3. Select **Chart** from bottom-left pane. Chart samples will be displayed.

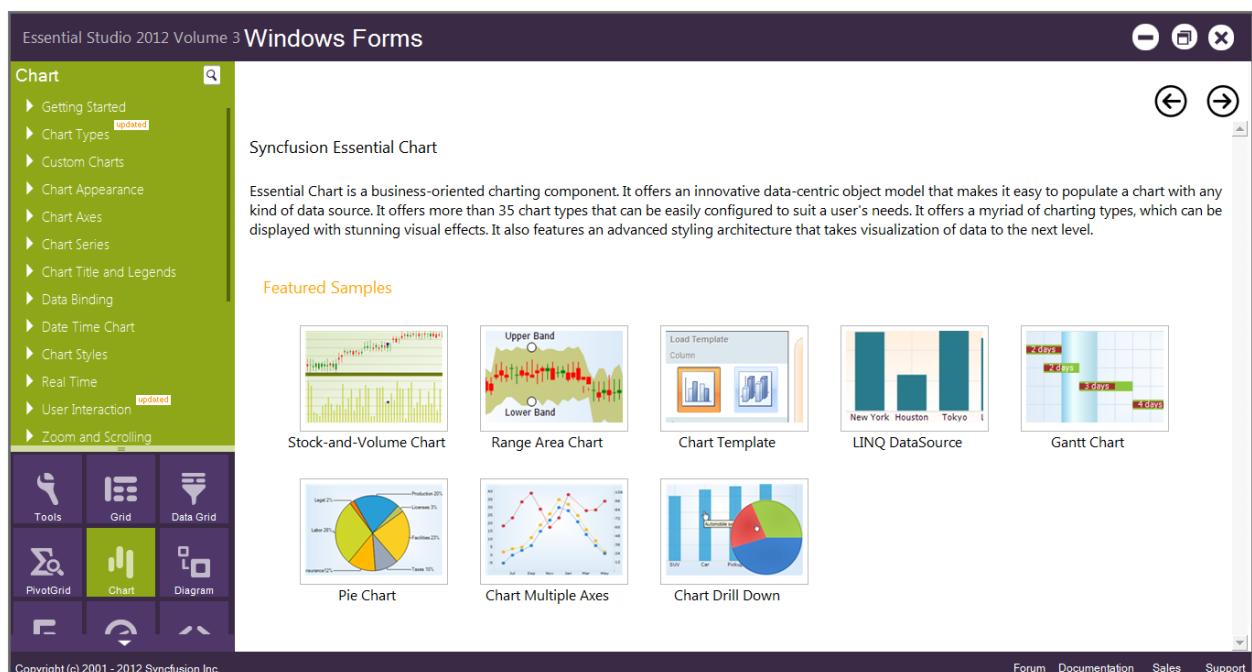


Figure 4: Chart Samples for Windows

4. Select any sample and browse through the features.

### **Source Code Location**

The default location of the Chart Windows source code is

[\[Install Drive\]:\Program Files\Syncfusion\Essential Studio\\[Version Number\]\Windows\Chart.Windows\Src](file:///C:/Program%20Files/Syncfusion/Essential%20Studio/15.1.0.34/Windows/Chart.Windows/Src)

## **2.3 Deployment Requirements**

### **Toolbox Entries**

- ChartControl
- Sparkline

### **Dll List**

While deploying an application that references Syncfusion Essential Chart assembly, the following dependencies must be included in the distribution.

3

- Syncfusion.Chart.Windows
- Syncfusion.Chart.Base
- Syncfusion.Shared.Base
- Syncfusion.Core

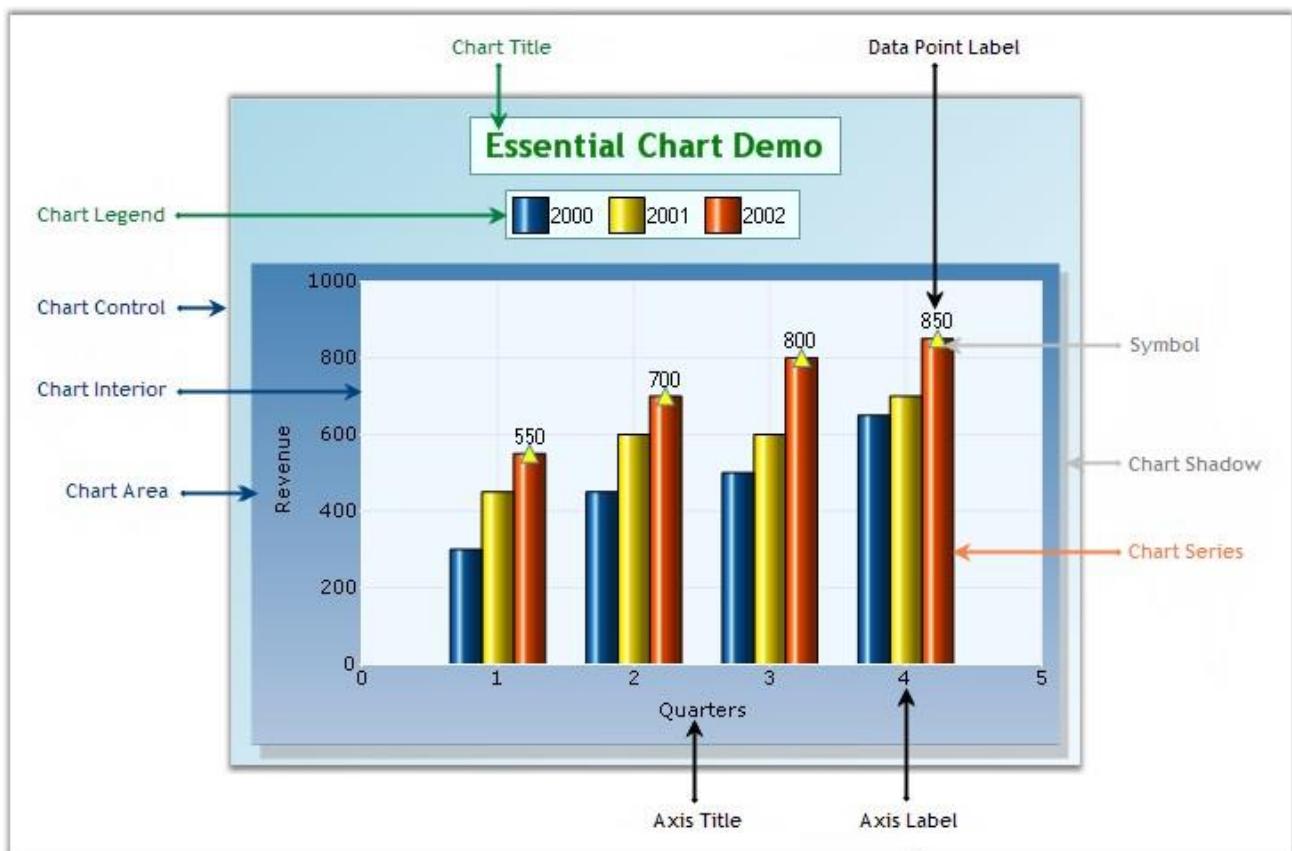
## 3 Getting Started

This section provides the details that you will need to know about getting started with our Chart control.

Here are the sub-topics covered under this section:

### 3.1 Control Structure

This section gives you an idea of the different sections of a Chart control. Below is the image that illustrates various sections of the control and their description in detail below.



*Figure 5: Chart Components*

## **Chart Title**

The Chart title is the area at the top of the Chart control that displays the text of the ChartControl.

## **Chart Legend**

The Chart legend is the portion of the display showing an entry for each of the data series added to the ChartControl. The Chart legend is positioned in line within the ChartControl (but outside the ChartArea) by default. However, if the chart legend is set to floating mode, the Chart legend can be positioned anywhere inside the Chart control.

The legend can also contain custom items with custom image and text.

## **Chart Area**

This is the section that holds the plots / graphs that are defined in the Chart. The Chart Area also includes the Chart axes that are defined. A primary X axis and primary Y axis are defined by default.

## **Chart control**

The ChartControl is the complete visible display for the Control. All other sections will be inside the ChartControl.

## **Chart Series**

Chart Series are the elements present inside the chart area.

## **Series Symbol**

Symbol that can be placed on the series points.

## **Chart Shadow**

Chart Area Shadow indicates the shadow of the chart area.

## **Chart Interior**

This section is the interior part of the chart control, within the chart area, which contains the data points plotted against X-axis and Y-axis.

### **Chart Label**

Represents the label texts on the axes data points in a Chart.

### **Axis Title**

Indicates the title for individual axis.

### **See Also**

[Text](#), [Chart Legend](#), [Chart Series](#), [Chart Area](#), [Axis Label Text Formatting](#), [Appearance and Positioning](#), [Customizing Label Text](#), [Intersecting Labels](#), [Grouping Labels](#)

## **3.2 Creating a simple Chart**

To create a simple chart control and populate it with simple data, follow the steps that are given below.

1. Open your form in the designer. Add the Syncfusion controls to your VS.NET toolbox if you haven't done so already (the install would have automatically done this unless you selected not to complete toolbox integration during installation).



*Figure 6: ChartControl in Toolbox*

2. Drag a Chart control onto the form.
3. Appearance and behavior-related aspects of the Chart can be controlled by setting the appropriate properties through the properties grid. For example, change the position of the legend for the control to be top aligned by changing the **LegendPosition** property.



*Figure 7: Setting Legend Position Property*

4. The data for the Chart can be added through code. Switch to code view in VS.NET and add the method shown below.

[C#]

```
using Syncfusion.Windows;
using Syncfusion.Windows.Forms.Chart;

// Create a new ChartSeries. The string specified is the name of the
// series.
ChartSeries series = this.chartControl1.Model.NewSeries("Series");

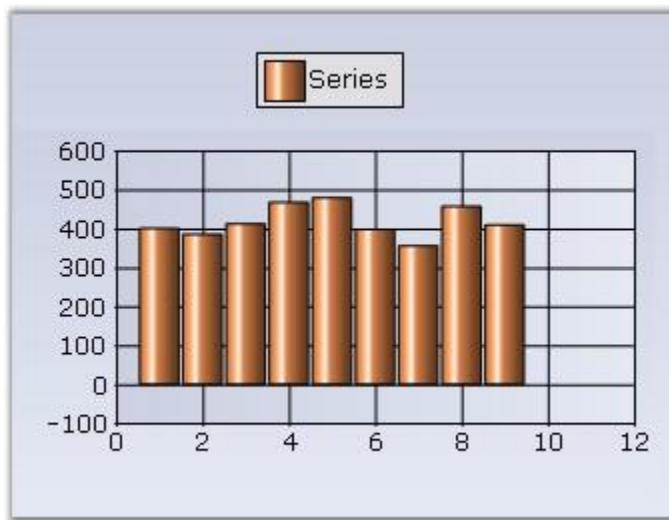
// Set the Text property of the series. This will be used by the legend
// to display a descriptive name.
series.Text = series.Name;
```

```
// Add points to the series.  
// format (x, y)  
series.Points.Add(1, 400);  
series.Points.Add(2, 385);  
series.Points.Add(3, 412);  
series.Points.Add(4, 467);  
series.Points.Add(5, 478);  
series.Points.Add(6, 397);  
series.Points.Add(7, 355);  
series.Points.Add(8, 456);  
series.Points.Add(9, 409);  
  
// Set the type of Chart.  
series.Type = ChartSeriesType.Column;  
  
// Add the series to the Chart.  
this.chartControl1.Series.Add(series);
```

**[VB.NET]**

```
Imports Syncfusion.Windows  
Imports Syncfusion.Windows.Forms.Chart  
  
' Create a new ChartSeries. The string specified is the name of the  
series.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series")  
  
' Set the Text property of the series. This will be used by the legend.  
series.Text = series.Name  
  
' Add points to the series.  
series.Points.Add(1, 400)  
series.Points.Add(2, 385)  
series.Points.Add(3, 412)  
series.Points.Add(4, 467)  
series.Points.Add(5, 478)  
series.Points.Add(6, 397)  
series.Points.Add(7, 355)  
series.Points.Add(8, 456)  
series.Points.Add(9, 409)  
  
' Set the type of Chart.  
series.Type = ChartSeriesType.Column  
  
' Add the series to the Chart.  
Me.chartControl1.Series.Add(series)
```

5. Now try running the project by selecting the Debug → Start Debugging. The chart will appear as shown below.



*Figure 8: Chart control*

#### **See Also**

[Binding a DataSet to a Chart](#), [Tasks Window](#)

## 4 Concepts and Features

---

This section will help you learn the core concepts and features of the Chart control. This will guide you to get started with the control. The topics provide complete support for the users, to implement the features available for this control, through the designer and also through programmatical approach. The following are the topics discussed.

### See Also

[Frequently Asked questions](#)

### 4.1 Chart Wizard

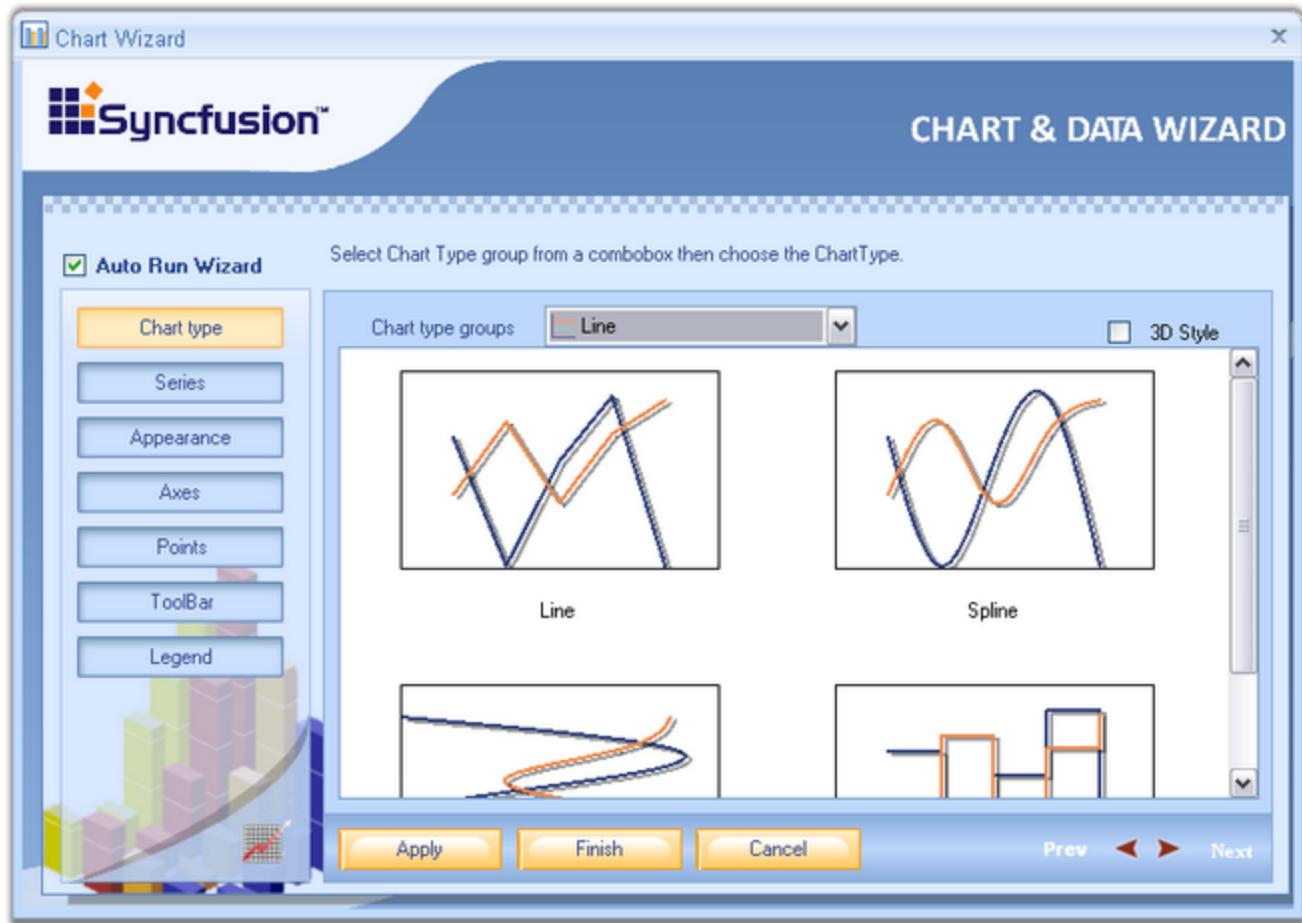
The Chart Wizard is a very convenient tool to setup the Chart during design-time.

The Wizard neatly categorizes the different portions of the Chart, and lets you customize the most common properties of these different portions easily.

#### **Key features of the Chart wizard**

1. Can create various types of chart.
2. Add series dynamically when the application is running.
3. Change the appearance of the chart with the various options that are provided to change the color palette, back color and title of the chart.
4. Customize the axes of the chart such as changing the range and labels.
5. Provides support for customization of the chart legend.
6. Customize the chart control's toolbar.
7. Lets you customize the point labels.

This section describes about the functionality of the chart wizard.



*Figure 9: Chart Wizard (Windows)*

### **Design Time**

To display the chart wizard at design-time, follow the below steps.

1. Add a ChartControl to your form.
2. Right-click anywhere in the chart to see a context menu.
3. Select the chart wizard item from the context menu.

### **At Run Time**

Optionally, you can also let your users to invoke this Wizard during run-time to let them customize the Chart's look and feel. To invoke the Chart wizard at runtime, use the below code.

[C#]

```
this.chartControl1.DisplayWizard();
```

[VB .NET]

```
Me.chartControl1.DisplayWizard()
```

The wizard provides six different categories whose settings can be customized.

1. [Chart type](#) to let you visualize and select the type of chart to display.
2. [Series](#) to let you add custom series to the chart and also setup data binding.
3. [Appearance](#) to customize the color, font etc. of the ChartControl and ChartArea.
4. [Axes](#) to change the chart control's axes settings.
5. [Points](#) to customize the point labels.
6. [Legend](#) to set the properties of the legend area.
7. [Toolbar](#) to customize the various properties of the toolbar.

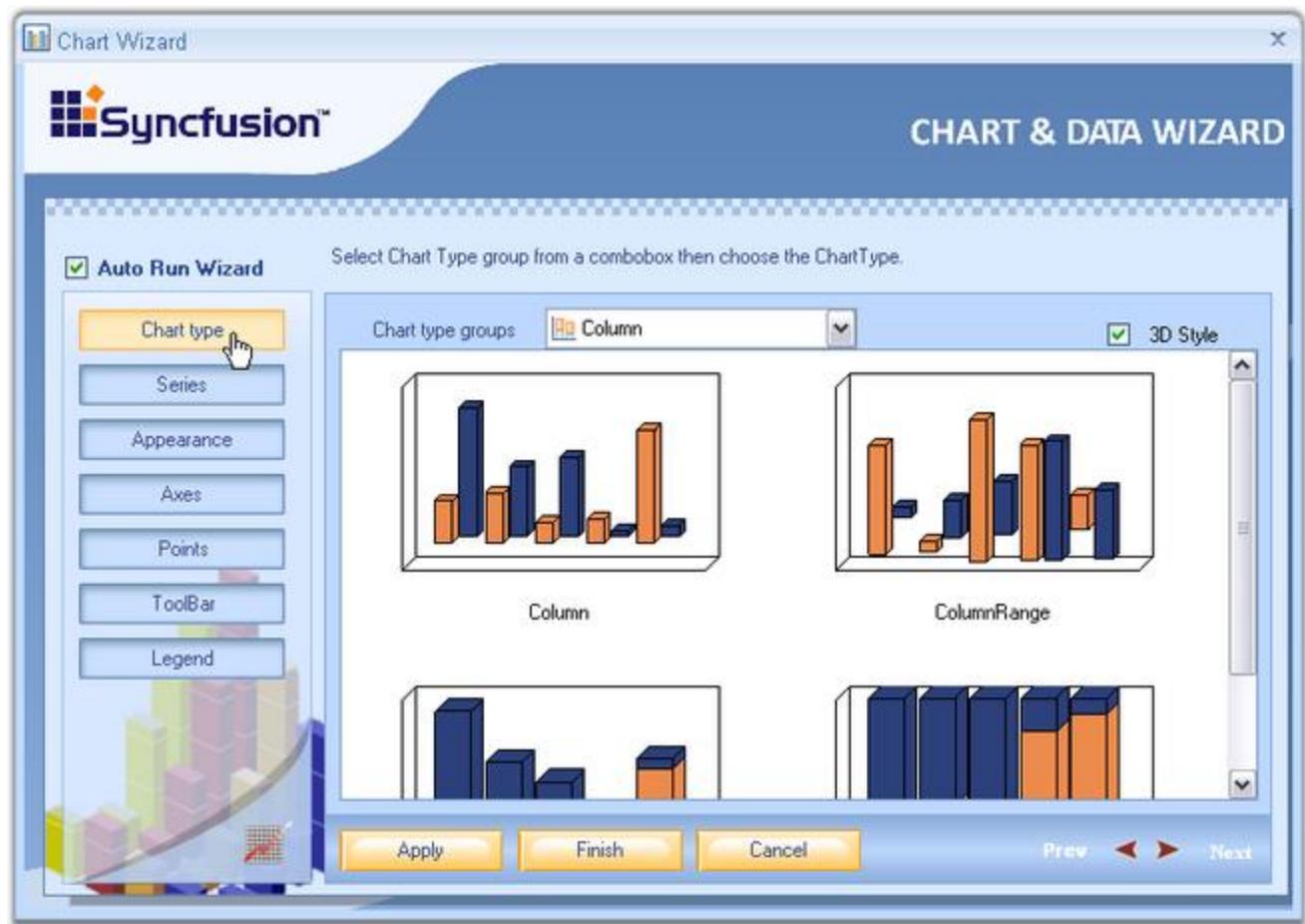
There is a preview panel where a Chart is rendered with the latest settings. The sub topics of this section will guide you through these settings.

After making necessary changes, click the **Apply** to apply those settings in the chart and finally, click the **Finish** to close the Wizard.

### 4.1.1 Chart Type

In this view, you can visually see the different chart types that could be used to render your data points. Simply click on the image of the chart type to pick the type you are interested in.

**3D Style** - The chart will be rendered in **3D** mode. Corresponds to the **Series3D** property.



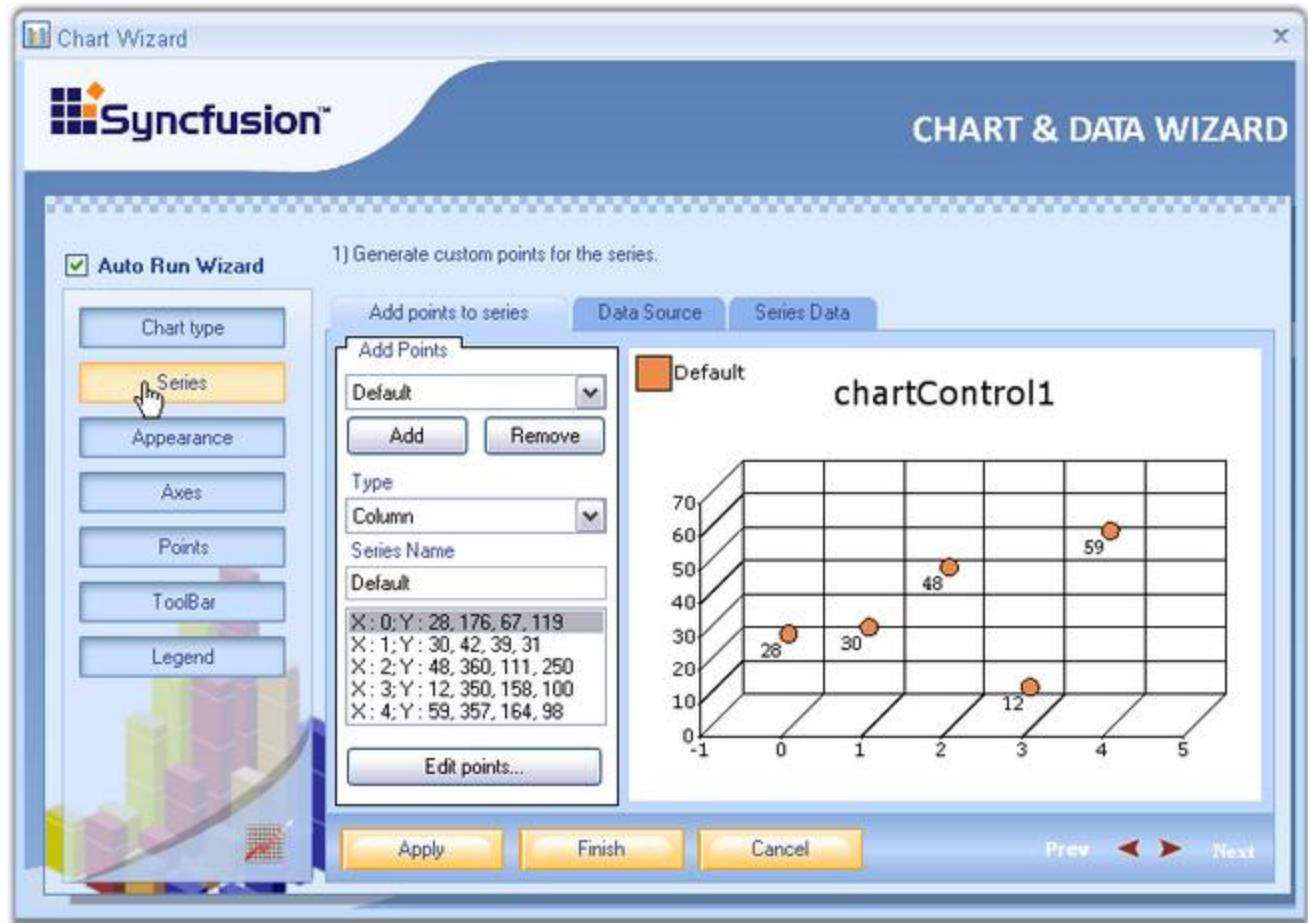
*Figure 10: Chart Type button selected in Chart Wizard*

#### **See Also**

[Chart Types](#)

#### **4.1.2 Series**

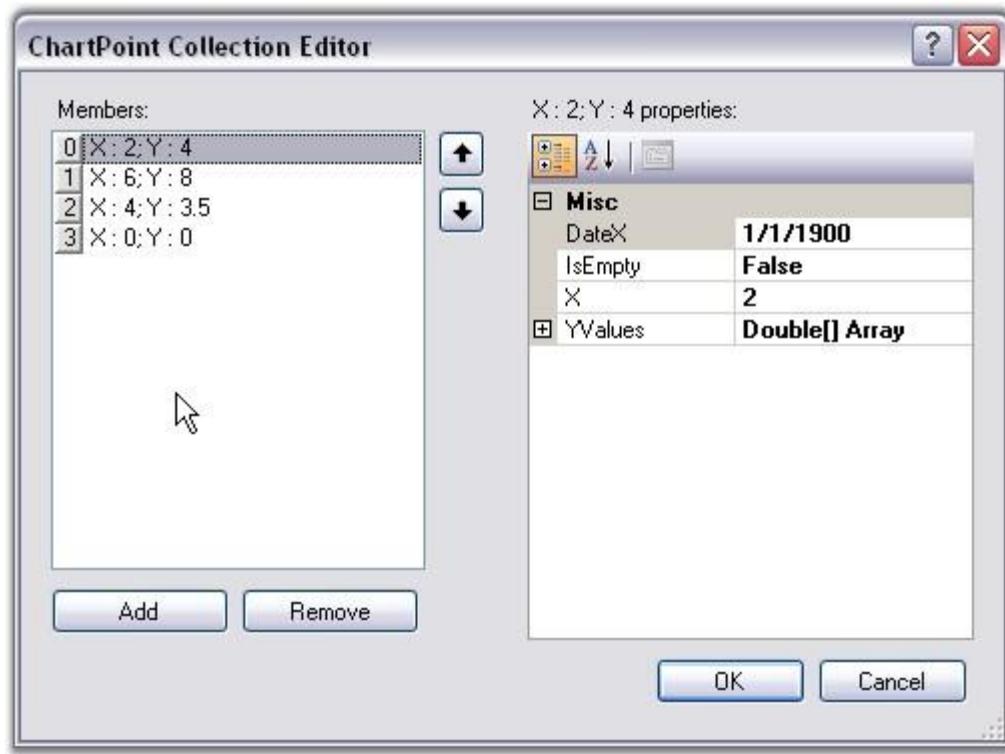
A Chart can display multiple series. Properties such as Name, Data source, Series Data can be set or changed for any of the series, easily, through this wizard.



*Figure 11: Series button selected in Chart Wizard*

Below are the three tabs in the Wizard for Series.

1. Add points to series
2. Click **Add** button to add a new series.
3. Select the series to which you need to add points.
4. This tab provides options to select any chart type using the **Type** combobox and series name using **Series Name** textbox.
5. Clicking the **Edit points** button, opens the **ChartPoint Collection Editor**.



*Figure 12: ChartPoint Collection Editor*

6. Click **Add** to add points to the series. Give **X** and **Y** values. Click **Ok**.

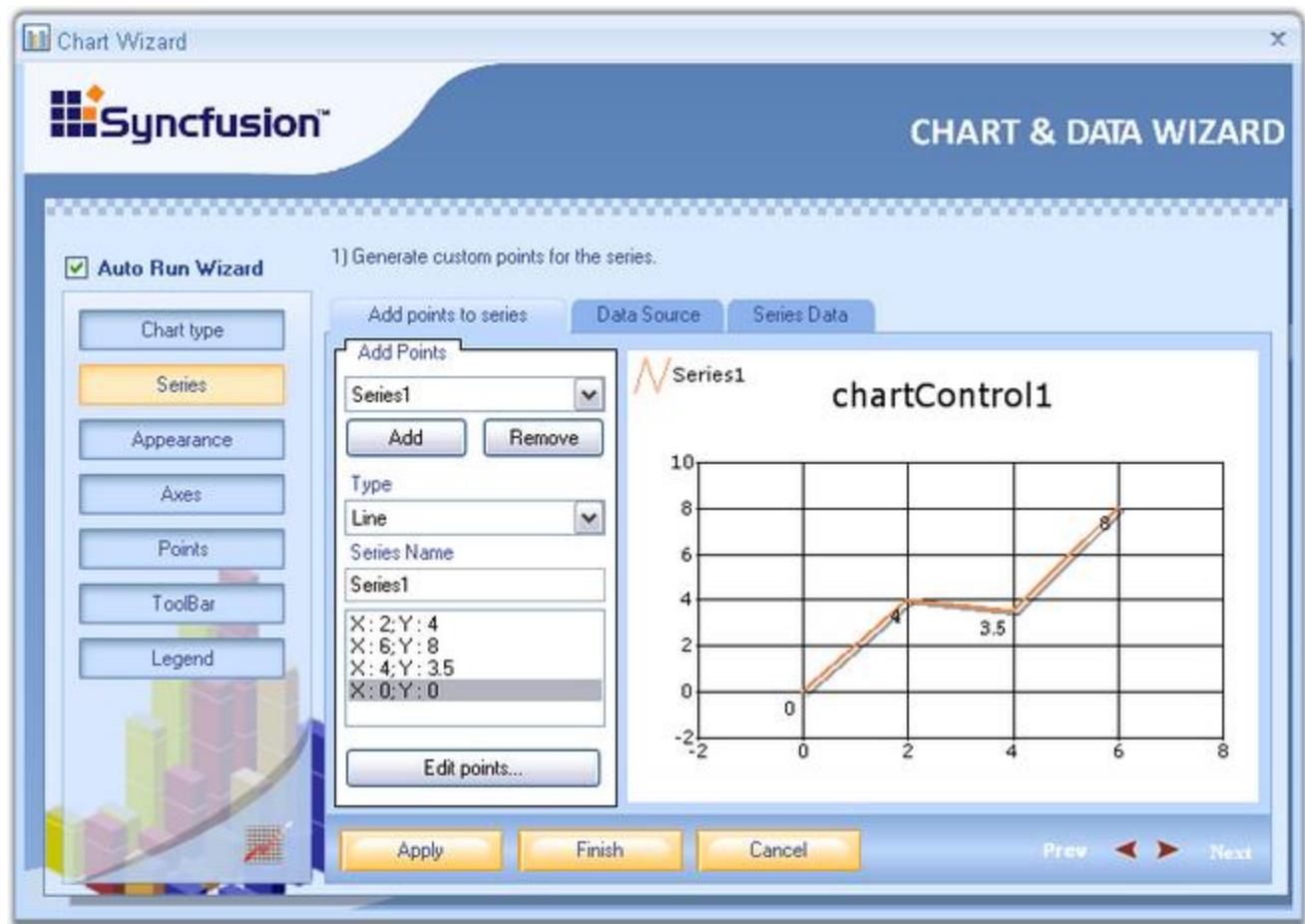
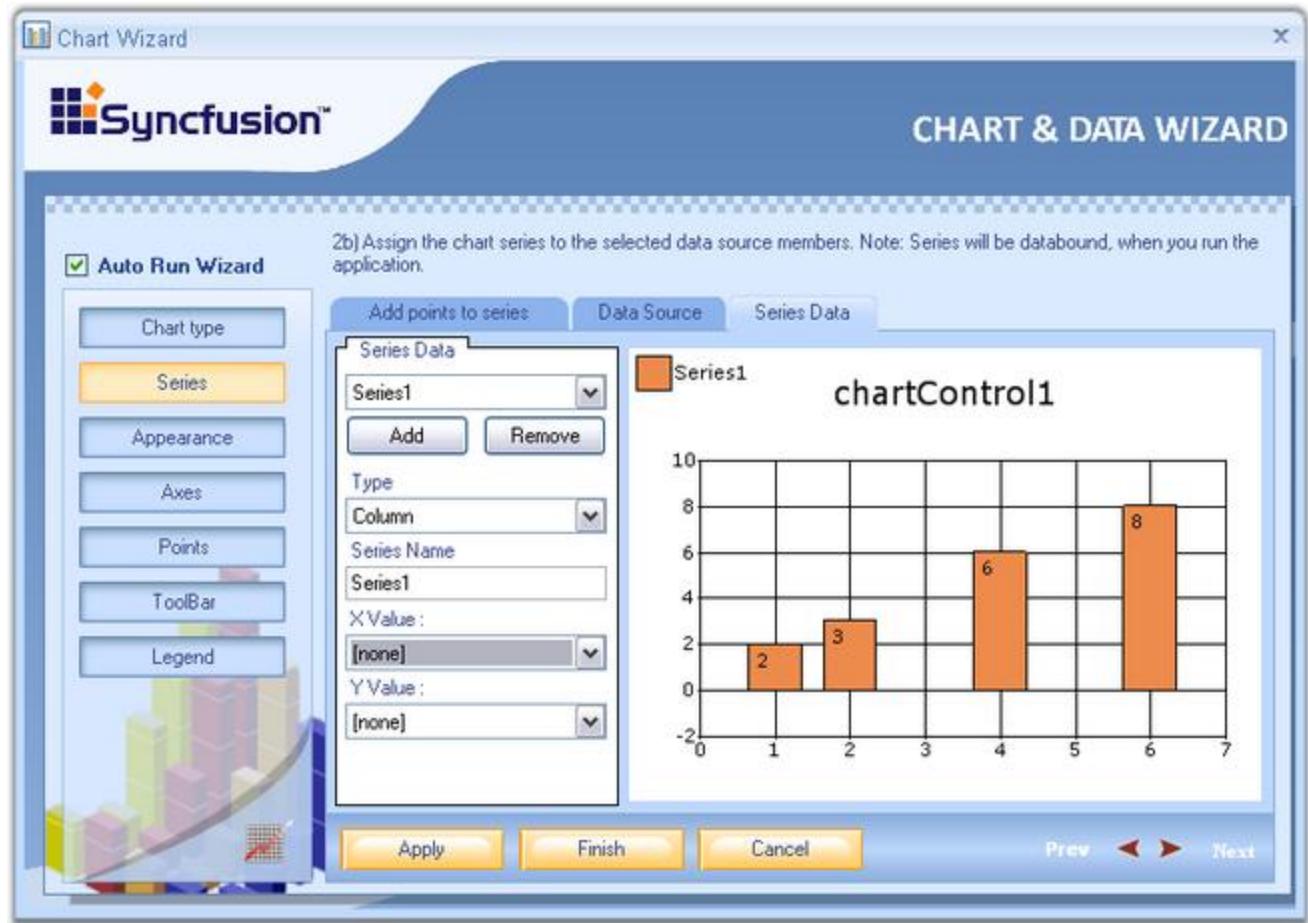


Figure 13: Adding Points to Series

**Data source** - The data source to connect with, can be selected using the data source page. Once the data source is selected, it will guide you through the connectivity steps. Refer [Data Binding in Chart Through Chart Wizard](#) topic for detailed information. This topic describes data binding techniques at the design-time through Chart Wizard tools.

**Series Data** - Using this tab, we can change the type of the chart. Whenever an external data source is selected using the Data Source tab, XValue and YValue ComboBox will be supplied with all the column names of the external data source.

Select one column for XValue and another for YValue, between which you wanted to draw the chart. Refer [Data Binding in Chart Through Chart Wizard](#) topic for detailed information.



*Figure 14: Setting the Chart Type*

#### See Also

[Chart Series](#)

### 4.1.3 Appearance

Customize the appearance of the ChartControl easily through the Chart Wizard. Here the appearance of the chart control and chart area such as color, font etc. can be set.

There are three tabs available in the wizard, for appearance settings.

- **Color Palette** - Essential Chart comes with a variety of built-in palettes. You can pick one of the palettes from the list and also get a preview of how the chart would look like. Note that you can also add a custom palette to the chart, but that has to be done in code.
- **Border and Back Color** - Under this tab, the back color for the chart control as well as for the chart area can be set. The user can also set the Border color and Border style for the chart area.
- **Title** - This gives options to set the title for the chart control. Also the position, alignment and color of the title can be set.

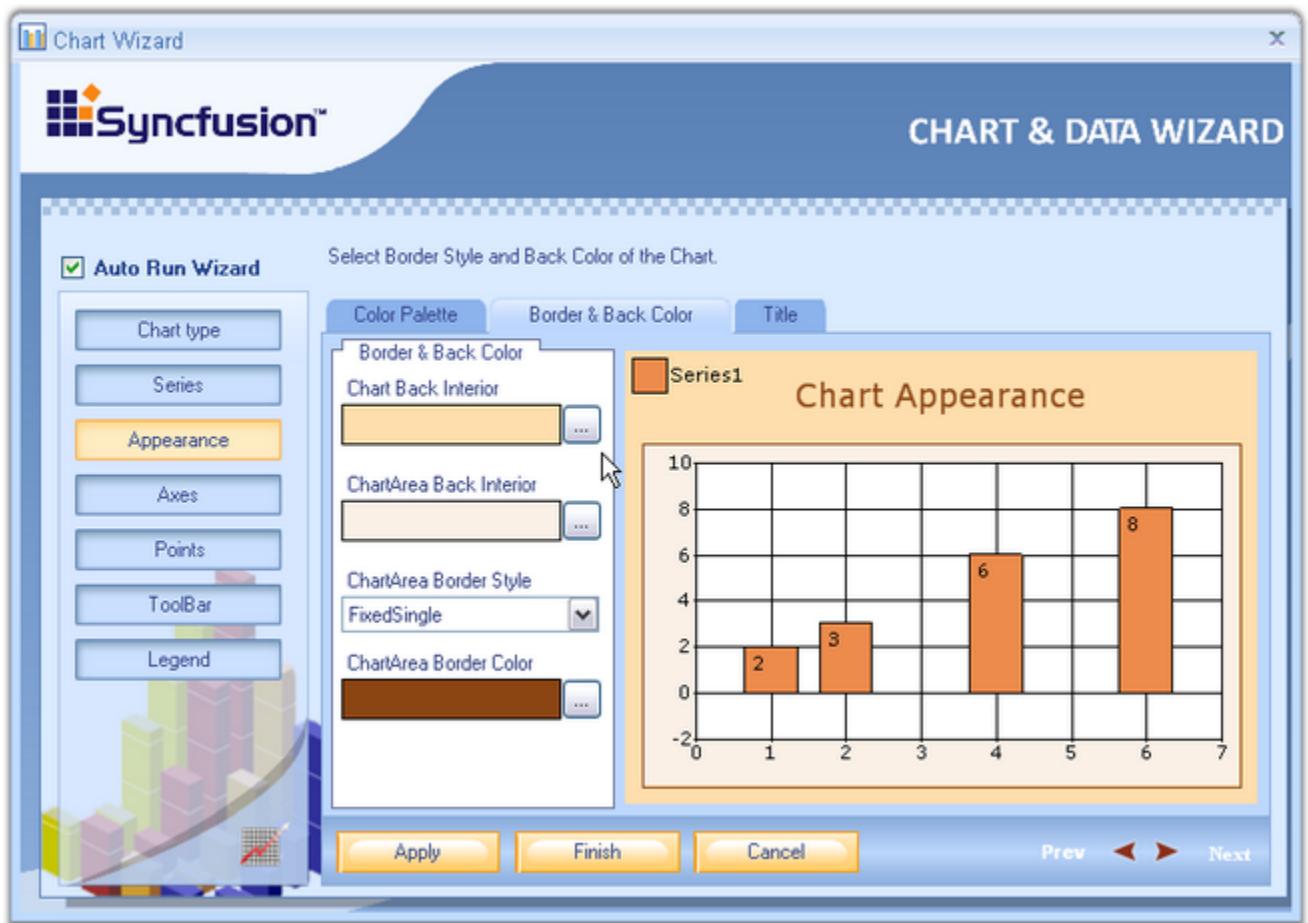


Figure 15: Appearance button selected in Chart Wizard

## See Also

[Appearance Settings](#)

#### 4.1.4 Axes

Various settings like grid line, axis title, value type, formats and other axes settings can be done using the wizard.

The below properties can be set separately for x-axis and y-axis.

- **Grid Lines** - Lets you show/hide the grid lines for this axis.
- **Axis Title** - The title text for the axis can be specified here.
- **Inversed, Opposed** - Specifies whether the axes are inversed, opposed.
- **Value Type** - If you know the type of data points you will be adding to this axis, specify it using the combo box. Possible value types are **double**, **datetime**, **custom** and **logarithmic**.
- **Format** - Specifies the label format.
- **Edit Labels** - The labels at the axes can be varied by entering the values in the **Collection Editor** Dialog box, which popup when the **Edit Labels** button is clicked.

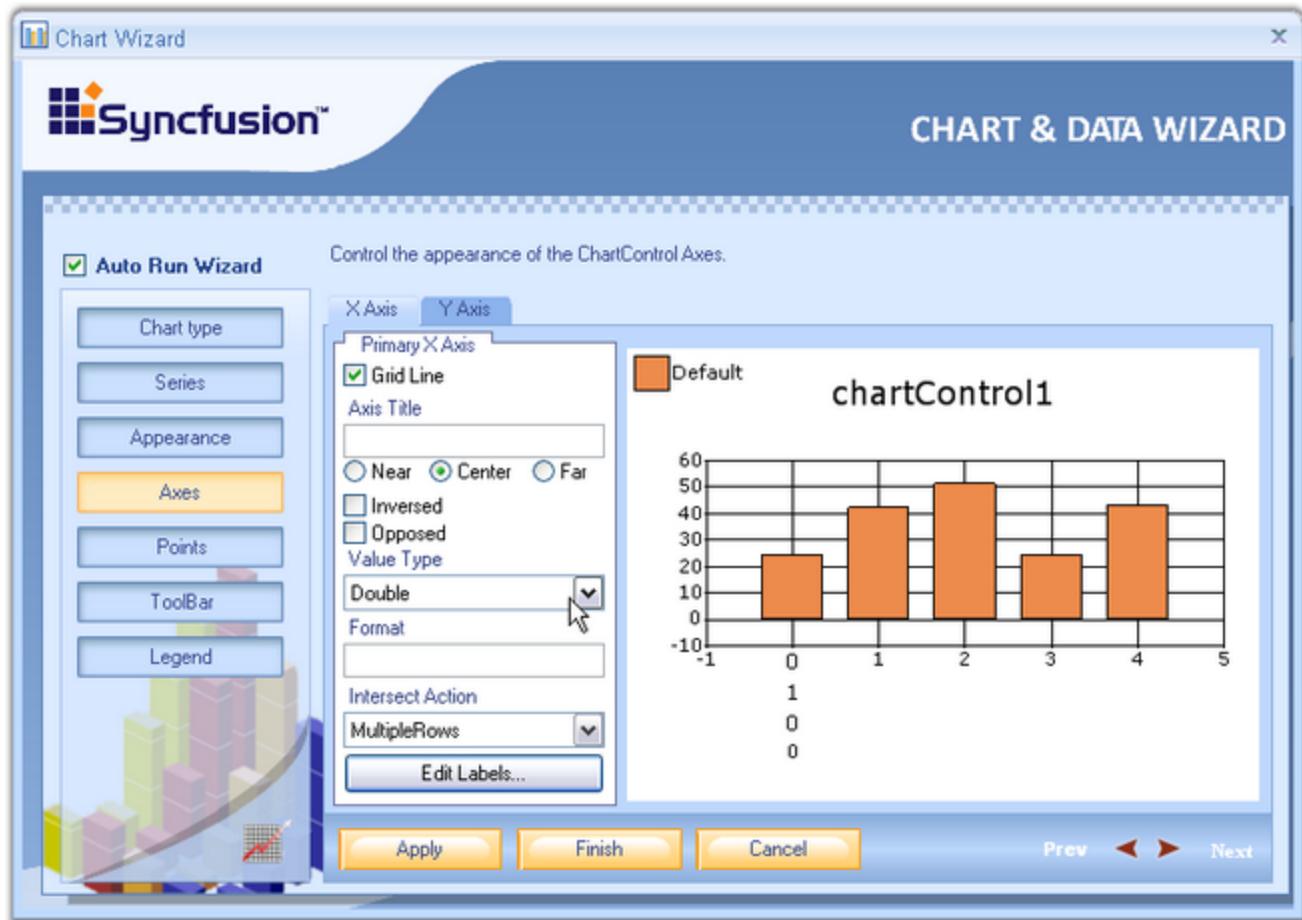


Figure 16: Axes button selected in Chart Wizard

- **Collection Editor Dialog** - Click the **Add** button to add a label to the collection. Select the added label to view its properties on its right side. The color of the label, font, value type and so on can be changed using the properties window. If any label needs to be removed from the collection editor, select the label and click the **Remove** button. After adding all the values to the collection and making the necessary changes in the properties, click **OK**. In a similar way the labels can be edited for y-axis also.

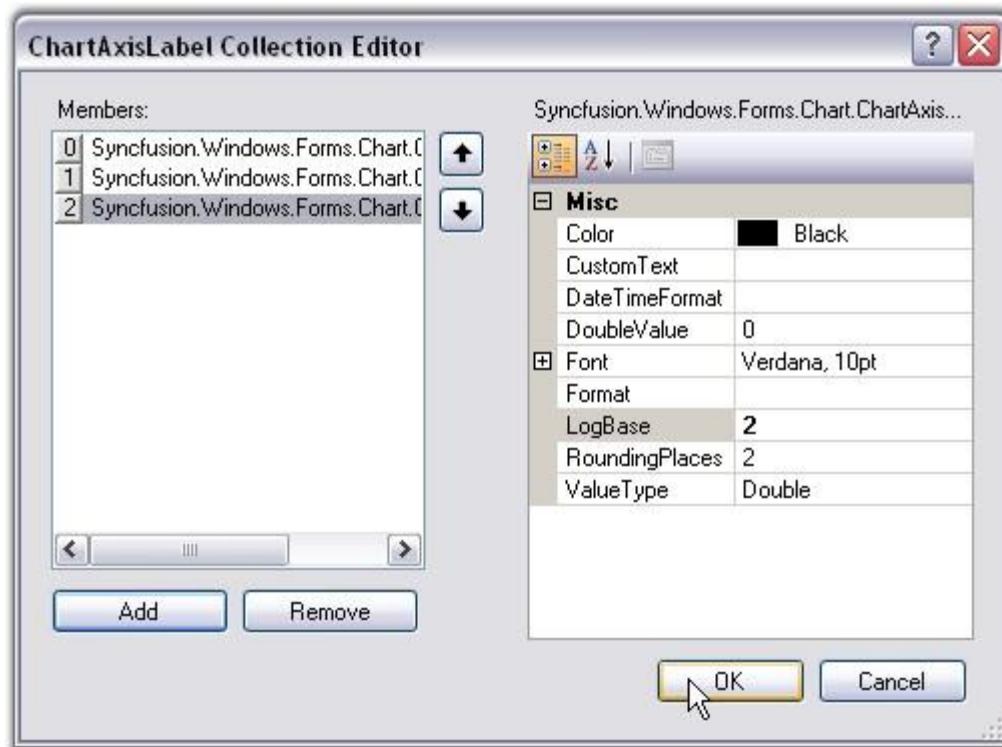


Figure 17: ChartAxisLabel Collection Editor

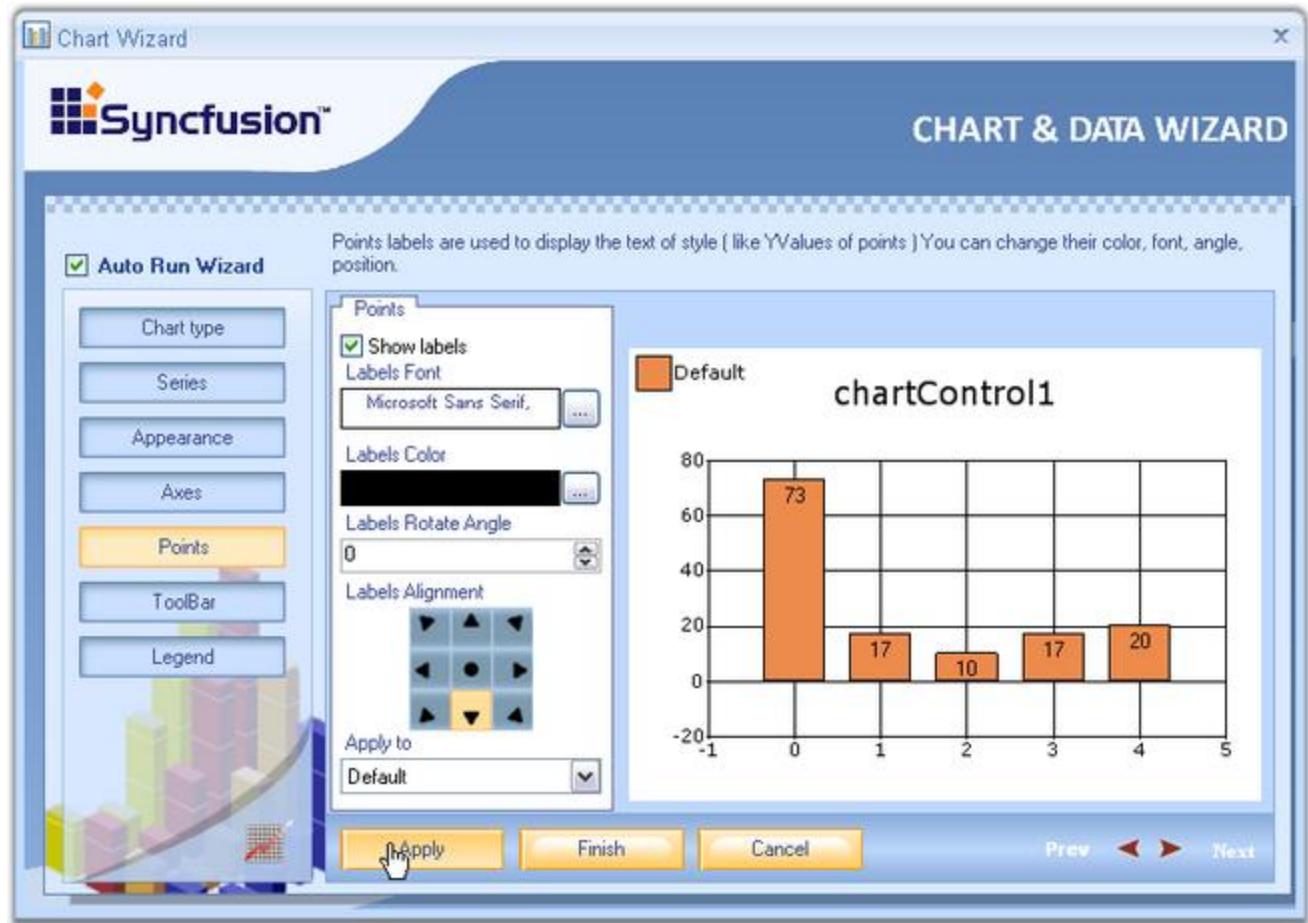
## See Also

[Chart Axes](#)

### 4.1.5 Points

This particular section in the wizard comes with the below options. It lets you display labels for the data points by simply checking **Show labels** option.

- **Labels Font** - Sets font style for the labels.
- **Labels Color** - Sets color for the labels.
- **Labels Rotate Angle** - Sets the angle of rotation of the labels.
- **Label Alignment** - Sets the alignment of the labels on the series points.
- **Apply To** - Specifies the series name to which the above settings should be applied.



*Figure 18: Points button selected in Chart Wizard*

#### 4.1.6 Toolbar

The final option in the Chart Wizard is the ChartControl ToolBar. It has two tabs.

- **ToolBar**

Under this tab, the user can customize the toolbar's back color, button style as well as set width and Height for the buttons through the respective options.

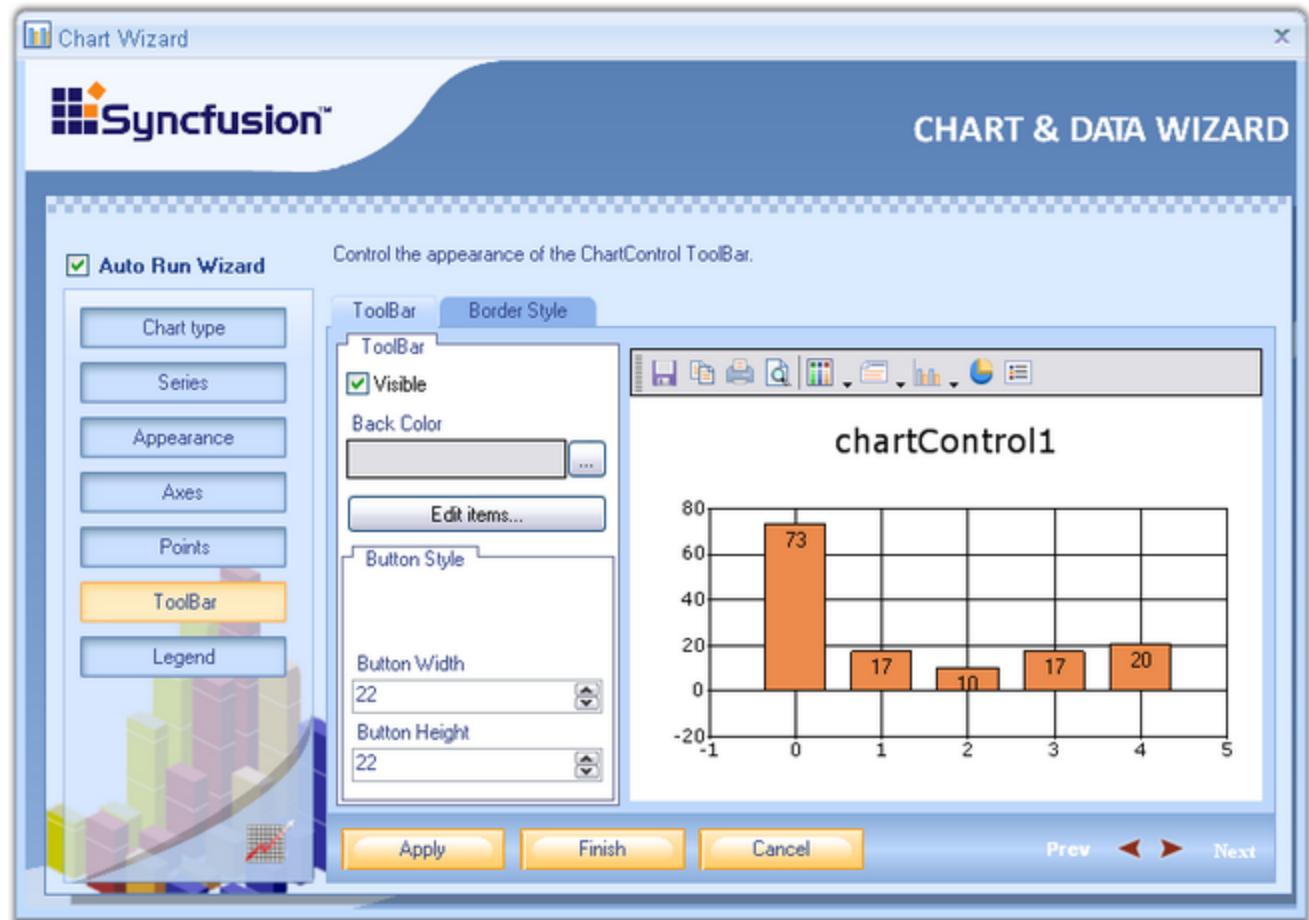
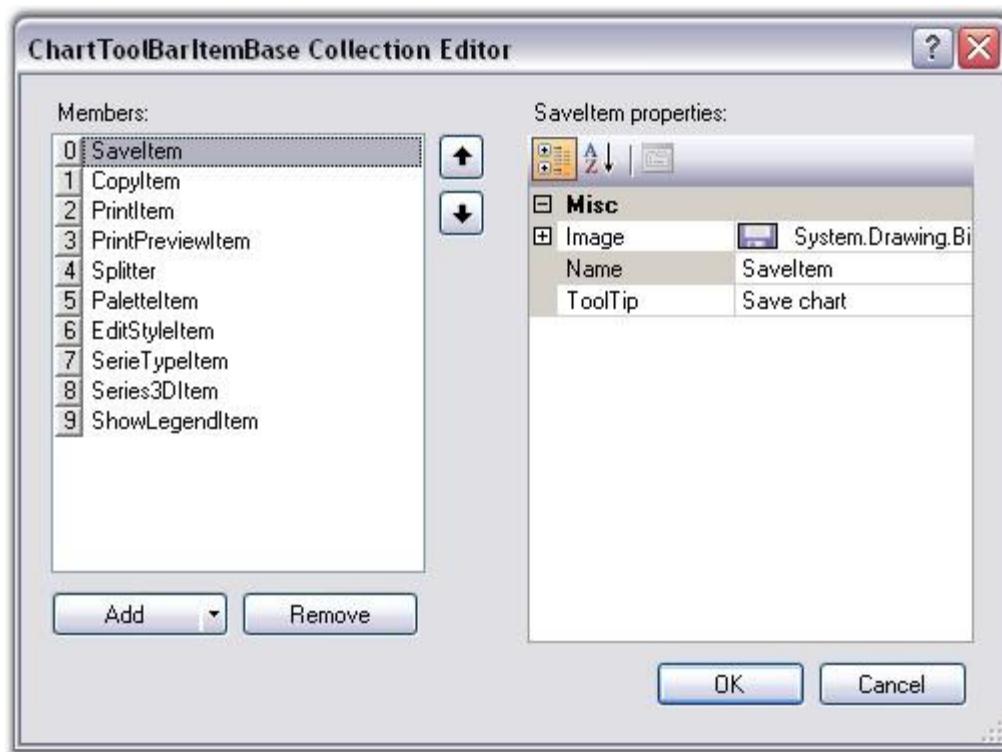


Figure 19: ToolBar button selected in ChartWizard

- **Edit Item**

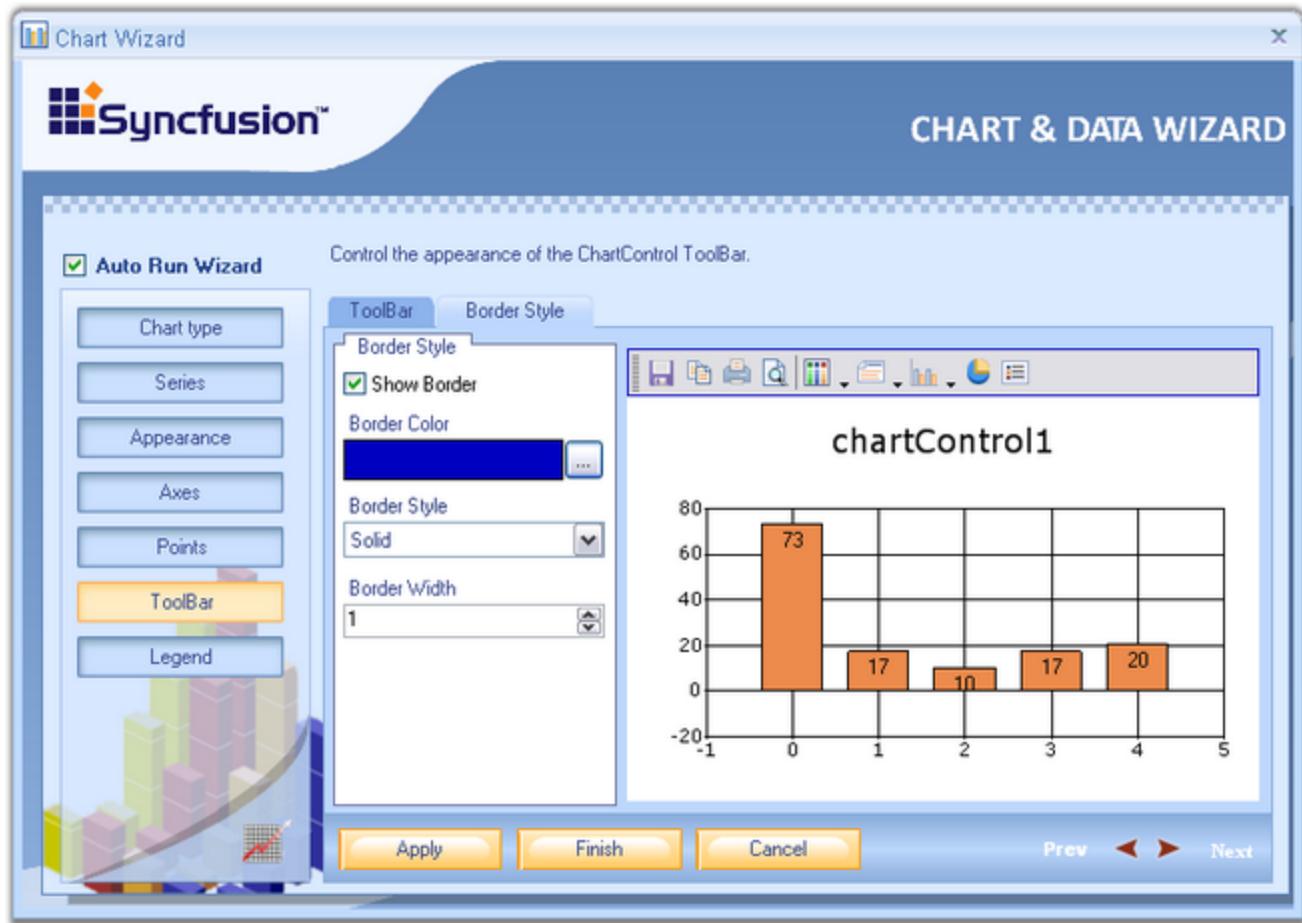
Clicking the Edit Items button will invoke the below editor. It provides options to change the image, name and tooltip for individual items.



*Figure 20: Editing ToolBar Items*

- **Border Style**

Toolbar's border, border style, border width and border color can be set through this tab.



*Figure 21: Customizing the Border Appearance of the ToolBar*

#### See Also

[Chart Toolbar](#)

#### 4.1.7 Legend

The various properties of the chart legend such as position, alignment, orientation etc., can be changed easily using this wizard. It has two tabs.

- **Legend** - In this window, the user can customize the legend's visibility, set its position and alignment using Position and Alignment combo boxes provided. You can also set back interior color using **Back Interior** combo box.

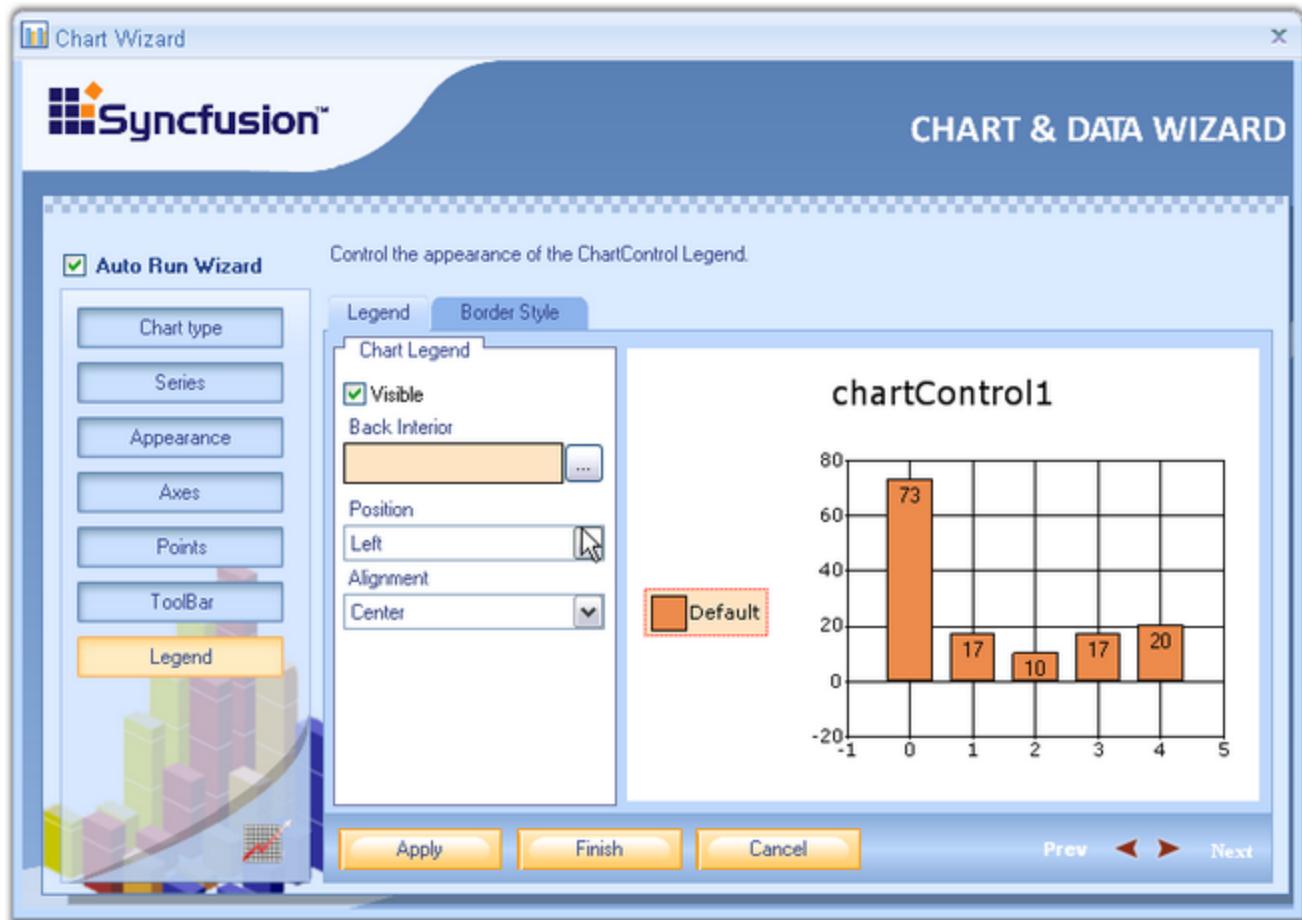


Figure 22: Legend button selected in Chart Wizard

- **Border Style** - Here the user can set custom borders to the legend. The visibility of the border, border color, its Dash style as well as the Border Width can be set easily by selecting the appropriate options.

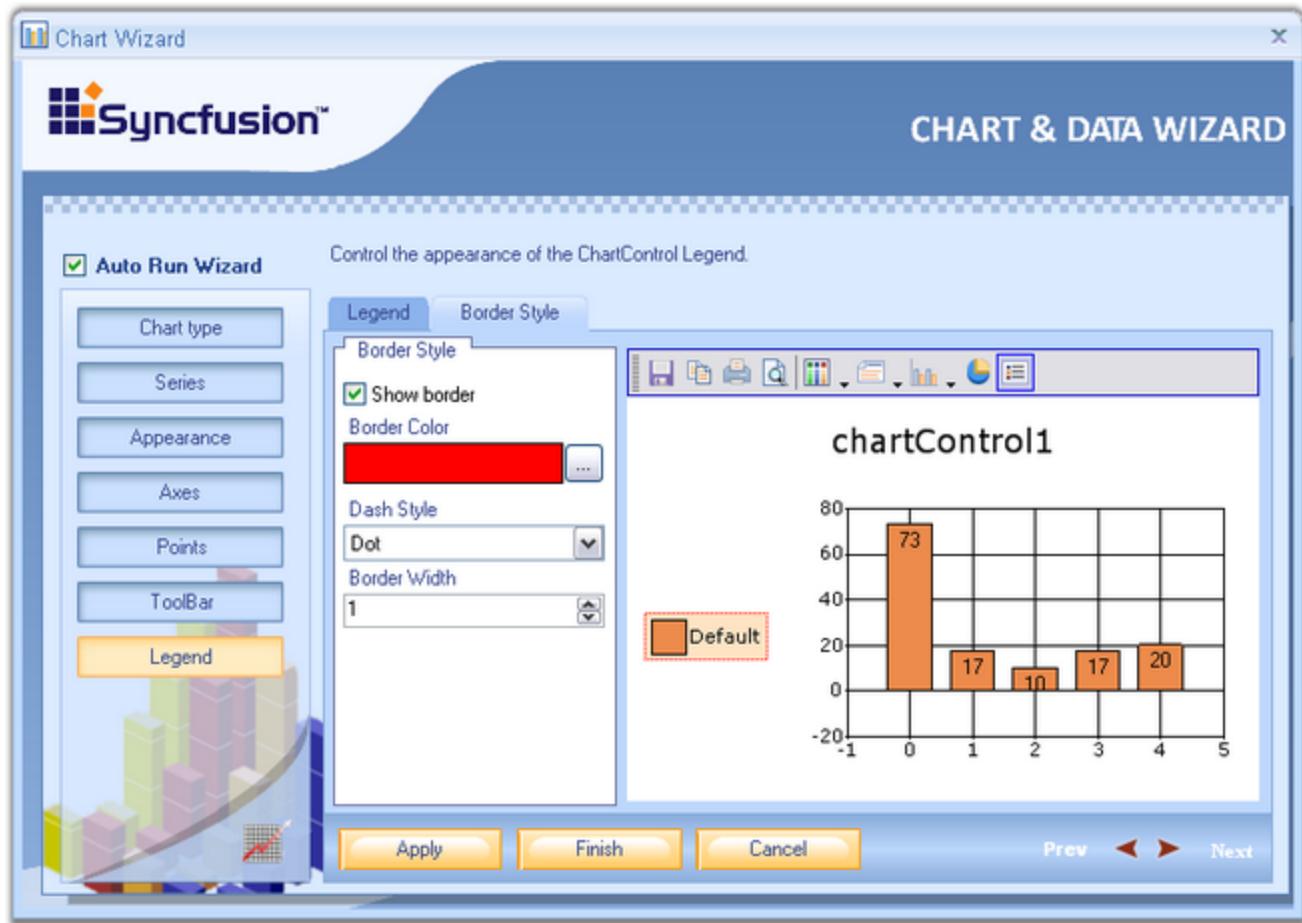


Figure 23: Customizing the Border Appearance of the Legend

## See Also

[Chart Legend](#)

## 4.2 Chart Data

### Built-in Support for data-binding

Essential Chart has built-in support for binding to  **DataTables**,  **DataSets**,  **DataViews** or any implementation of  **IListSource**,  **IBindingList** or  **ITypedList**.

The ChartSeries data points and the axis labels are the ones that can be databound.

There is however no DESIGN-TIME support for data binding. This has to be setup in code.

[Binding a DataSet to the Chart](#)

### Data binding via custom interfaces

There is also a more flexible support for implementing custom data models by implementing specific interfaces. Using this approach you can query and provide data for the chart much more flexibly and from any kind of data store.



**Note:** One important reason you might want to use either of the above two approaches is to greatly enhance performance (speed and memory) especially while dealing with a large set of data points.

#### See Also

[Implementing Custom Data Binding Interfaces](#)

### 4.2.1 Binding a DataSet to the Chart

The following sample code illustrates how a custom **DataSet** can be bound to a **ChartSeries** to provide data points and to a **ChartAxis** to provide label names. Note that the DataSet can easily be replaced with a **DataTable** or **DataView**.

Demographics...hartData.mdb)			
	ID	City	Population
	1	New York	13
	2	Houston	6
	3	Tokyo	17
	4	London	15
	5	Los Angeles	11

Figure 24: Access Table data that is about to get bound to Chart

[C#]

```
ChartDataBindModel model = null;
ChartDataBindAxisLabelModel xAxisLabelModel = null;

// A custom DataSet bound to the Demographics table
private ChartAccessDataBind.DataSet1 dataSet11;

this.oleDbTypeAdapter1.Fill(this.dataSet11.Demographics);

model = new ChartDataBindModel(this.dataSet11, "Demographics");
// The column that contains the X values.
model.XName = "ID";
// The columns that contain the Y values.
model.YNames = new string[] {"Population"};

ChartSeries series = this.chartControl1.Model.NewSeries("Data Bound Series");
series.Text = series.Name;
series.SeriesModelImpl = model;

series.Style.TextColor = Color.White;
series.Style.Font.Bold = true;

this.chartControl1.Series.Add(series);

this.xAxisLabelModel = new
ChartDataBindAxisLabelModel(this.dataSet11, "Demographics");
// The columns that has the label values for the corresponding X values.
this.xAxisLabelModel.LabelName = "City";

this.chartControl1.PrimaryXAxis.LabelsImpl = this.xAxisLabelModel;
this.chartControl1.PrimaryXAxis.ValueType = ChartValueType.Custom;
```

**[VB.NET]**

```
Dim model As ChartDataBindModel = Nothing
Dim xAxisLabelModel As ChartDataBindAxisLabelModel = Nothing

' A custom DataSet bound to the Demographics table
Private dataSet11 As ChartAccessDataBind.DataSet1

Me.oleDbTypeAdapter1.Fill(Me.dataSet11.Demographics)

model = New ChartDataBindModel(Me.dataSet11, "Demographics")
' The column that contains the X values.
```

```
model.XName = "ID"
' The columns that contain the Y values.
model.YNames = New String() {"Population"}

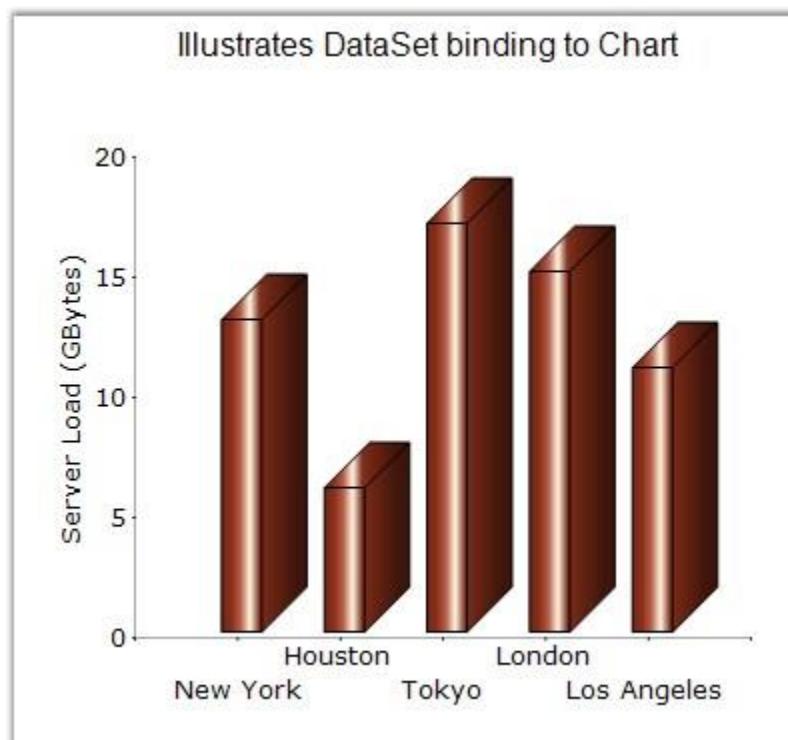
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Data
Bound Series")
series.Text = series.Name
series.SeriesModelImpl = model

series.Style.TextColor = Color.White
series.Style.Font.Bold = True

Me.chartControl1.Series.Add(series)

Me.xAxisLabelModel = New
ChartDataBindAxisLabelModel(Me.dataSet11, "Demographics")
' The columns that has the label values for the corresponding X values.
Me.xAxisLabelModel.LabelName = "City"

Me.chartControl1.PrimaryXAxis.LabelsImpl = Me.xAxisLabelModel
Me.chartControl1.PrimaryXAxis.ValueType = ChartValueType.Custom
```



*Figure 25: Demographics DataSet bounded to the Chart*

## 4.2.2 Implementing Custom Data Binding Interfaces

Note that the **ChartDataBindModel** type in the previous topic implements a simple interface called **IChartSeriesModel**. This interface requires the implementation of one property, two methods and one optional event. So, you can easily provide a custom implementation of this interface instead of using the ChartDataBindModel.

Shown below is some sample code that implements IChartSeriesModel interface for use with the chart.

[C#]

```
// Custom Model
public class ArrayModel : IChartSeriesModel
{
    private double[] data;

    public ArrayModel(double[] data)
    {
        this.data = data;
    }

    // Returns the number of points in this series.
    public int Count
    {
        get
        {
            return this.data.GetLength(0);
        }
    }

    // Returns the Y value of the series at the specified point index.
    public double[] GetY(int xIndex)
    {
        return new double[] {data[xIndex]};
    }

    // Returns the X value of the series at the specified point index.
    public double GetX(int xIndex)
    {
```

```
        return xIndex;
    }

    // Indicates whether a plottable value is present at the specified
    point index.
    public bool GetEmpty(int index)
    {
        return false;
    }

    // Event that should be raised by any implementation of this interface
    if data that it holds changes. This will cause the chart to be updated
    accordingly. We don't raise this event in our implementation as our
    data will be static.
    public event ListChangedEventHandler Changed;
}
```

**[VB.NET]**

```
' Custom Model
Public Class ArrayModel Implements IChartSeriesModel
    Private data() As Double

    Public Sub New(ByVal data() As Double)
        Me.data = data
    End Sub

    ' Returns the number of points in this series.
    Public ReadOnly Property Count() As Integer
        Get
            Return Me.data.GetLength(0)
        End Get
    End Property

    ' Returns the Y value of the series at the specified point index.
    Public Double() GetY(Integer xIndex)
        Return New Double(){data(xIndex)}
    End Function

    ' Returns the X value of the series at the specified point index.
    Public Double GetX(Integer xIndex)
        Return xIndex
    End Function

    ' Indicates whether a plottable value is present at the specified
    point index.
```

```
Public Function GetEmpty(ByVal index As Integer) As Boolean
    Return False
End Function

' Event that should be raised by any implementation of this
interface if data that it holds changes. This will cause the chart to
be updated accordingly. We don't raise this event in our implementation
as our data will be static.
Public event ListChangedEventHandler Changed
End Class
```

Bind the above model to the chart series.

**[C#]**

```
//Creating series data and binding to the array model
ChartSeries series1 = this.chartControl1.Model.NewSeries("Series 1");
series1.SeriesIndexedModelImpl = new ArrayModel(new
double[]{22,24,32,12,18});
series1.Type = ChartSeriesType.Bar;
this.chartControl1.Series.Add(series1);
```

**[VB .NET]**

```
'Creating series data and binding to the array model
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("Series
1")
series1.SeriesIndexedModelImpl = New ArrayModel(New
Double(){22,24,32,12,18})
series1.Type = ChartSeriesType.Bar
Me.chartControl1.Series.Add(series1)
```



*Figure 26: CustomModel bound to the Chart Series*

#### **Indexed data**

Note that if you have indexed data, which implies that the X values are simply categories and don't carry any cardinal value, then you can instead implement the **IChartSeriesIndexedModel** interface and bind it to the **ChartSeries.SeriesIndexedModelImpl**. The main difference in this interface is that you don't have to implement the **GetX** method.

### **4.2.3 Chart Data Binding with IEnumerables**

Syncfusion chart provides an option of binding the Chart with **IEnumerables**, like **ArrayList** for Indexed or Non Indexed model data through **ChartDataBindModel** implementation.

[C#]

```
class PopulationData
```

```
{  
  
    private string city;  
  
    public string City  
    {  
  
        get { return city; }  
  
        set { city = value; }  
    }  
  
    private double population;  
  
    public double Population  
    {  
  
        get { return population; }  
  
        set { population = value; }  
    }  
  
    public PopulationData(string city, double population)  
    {  
        this.city = city;  
        this.population = population;  
    }  
}
```

**[VB.NET]**

```
Class PopulationData  
  
    Private m_city As String  
  
    Public Property City() As String  
  
        Get  
            Return m_city  
        End Get  
  
        Set(ByVal value As String)  
            m_city = value
```

```
    End Set
End Property

Private m_population As Double

Public Property Population() As Double

    Get
        Return m_population
    End Get

    Set(ByVal value As Double)
        m_population = value
    End Set
End Property
```

If you have a class like above, you will have a collection of this class instances, in an ArrayList. To bind with the Chart, you need to create a ChartDataBindModel instance, by supplying the instance of data source(In our case, ArrayList is the data source).

In this example, we are binding with a Non Indexed data, with YNames alone and the chart will not be rendered with x-axis values. We need to assign the x-axis values through **ChartDataBindAxisLabelModel** class. ChartDataBindAxisLabelModel class provides a facility to bind the axis label values through the data source like ChartDataBindModel.

[C#]

```
ArrayList populations = new ArrayList();
populations.Add(new PopulationData("New York", 13));
populations.Add(new PopulationData("Houston", 6));
populations.Add(new PopulationData("Tokyo", 17));
populations.Add(new PopulationData("London", 15));
populations.Add(new PopulationData("Los Angels", 11));

ChartSeries series = new ChartSeries("Populations");
ChartDataBindModel dataSeriesModel = new
ChartDataBindModel(populations);

// If ChartDataBindModel.XName is empty or null, X value is index of
// point.
//Here I have assigned the property name Population as Y axis name and
ChartDataBindModel automatically detects the Population property and
will bind the data from it.
dataSeriesModel.YNames = new string[] { "Population" };
```

```
//Binding the ChartDataBindModel with the Series. This is the best
practise for binding with the large amount of data since it will reduce
the performance issue of Chart rendering and manipulating data.
series.SeriesModel = dataSeriesModel;

//Since we have specified YNames only for the DataBind model, it will
take the data source is non indexed model and it will ignore the X axis
values. We need to assign the X axis values what we need to show on X
axis by ChartDataBindAxisLabelModel separately.
ChartDataBindAxisLabelModel dataLabelsModel = new
ChartDataBindAxisLabelModel(populations);

dataLabelsModel.LabelName = "City";

chartControl1.Series.Add(series);
chartControl1.PrimaryXAxis.LabelsImpl = dataLabelsModel;
```

**[VB.NET]**

```
Dim populations As New ArrayList()
populations.Add(New PopulationData("New York", 13))
populations.Add(New PopulationData("Houston", 6))
populations.Add(New PopulationData("Tokyo", 17))
populations.Add(New PopulationData("London", 15))
populations.Add(New PopulationData("Los Angeles", 11))

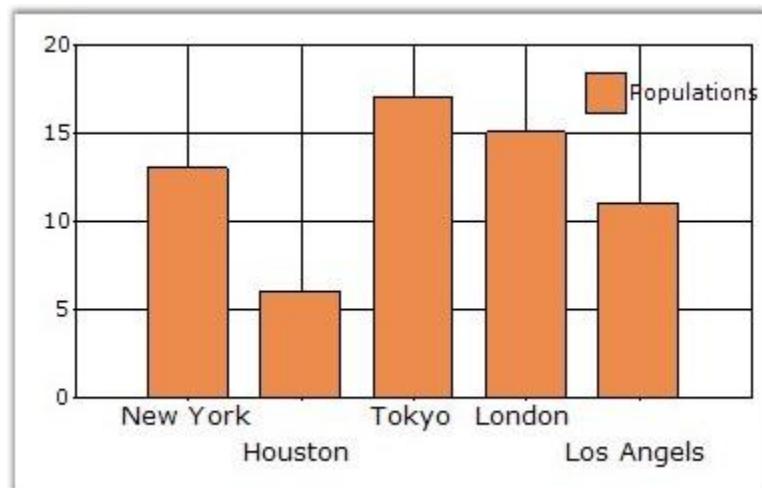
Dim series As New ChartSeries("Populations")
Dim dataSeriesModel As New ChartDataBindModel(populations)

' If ChartDataBindModel.XName is empty or null, X value is index of
point.
'Here I have assigned the property name Population as Y axis name and
ChartDataBindModel automatically detects the Population property and
will bind the data from it.
dataSeriesModel.YNames = New String() {"Population"}

'Binding the ChartDataBindModel with the Series. This is the best
practise for binding with the large amount of data since it will reduce
the performance issue of Chart rendering and manipulating data.
series.SeriesModel = dataSeriesModel

'Since we have specified YNames only for the DataBind model, it will
take the data source is non indexed model and it will ignore the X axis
values. We need to assign the X axis values what we need to show on X
axis by ChartDataBindAxisLabelModel separately.
```

```
Dim dataLabelsModel As New ChartDataBindAxisLabelModel(populations)  
  
dataLabelsModel.LabelName = "City"  
  
chartControl1.Series.Add(series)  
chartControl1.PrimaryXAxis.LabelsImpl = dataLabelsModel
```



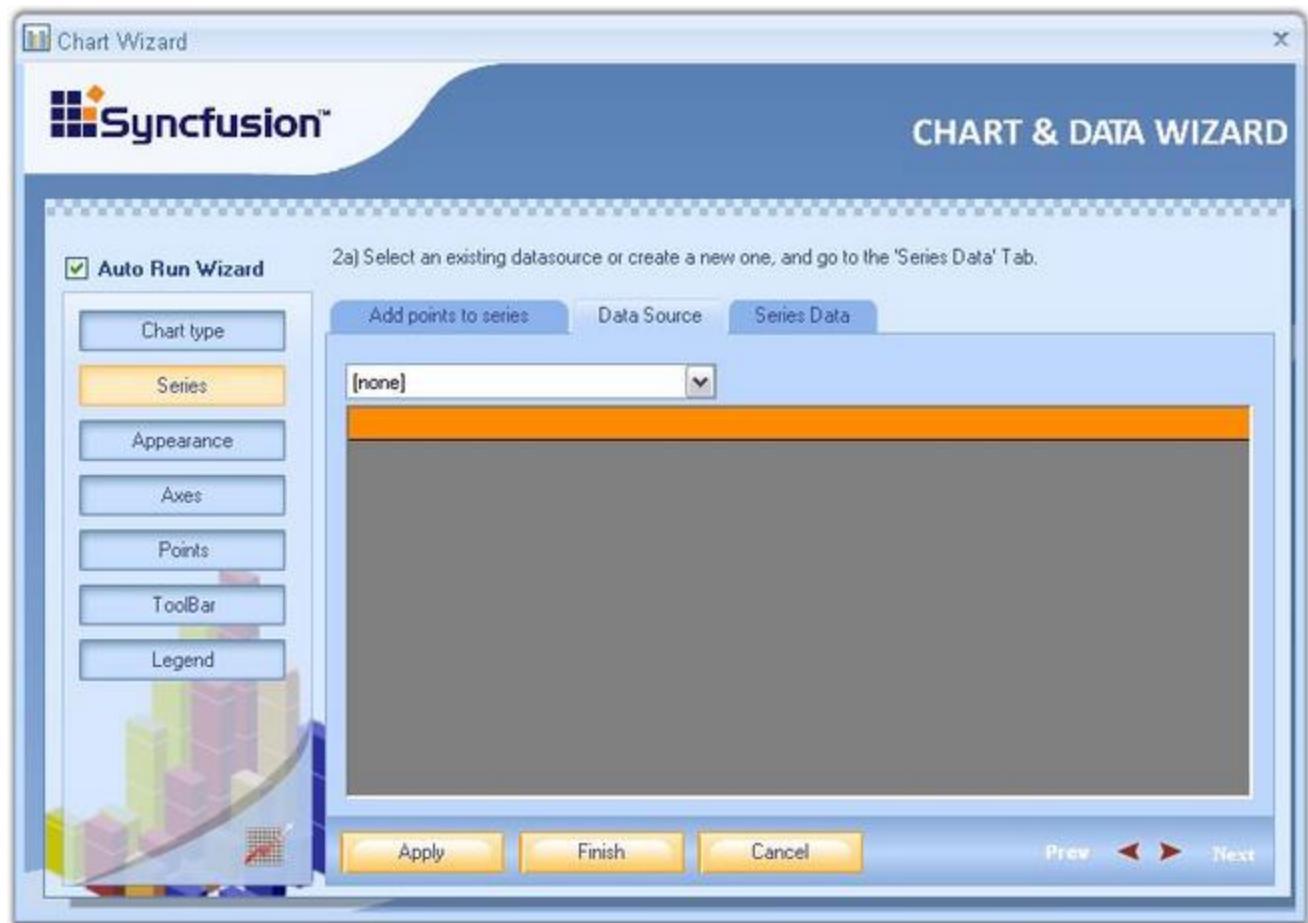
*Figure 27: Binding Chart with IEnumerables*

#### 4.2.4 Data Binding in Chart Through Chart Wizard

You can easily implement data binding technique at design-time, using Chart Wizard.

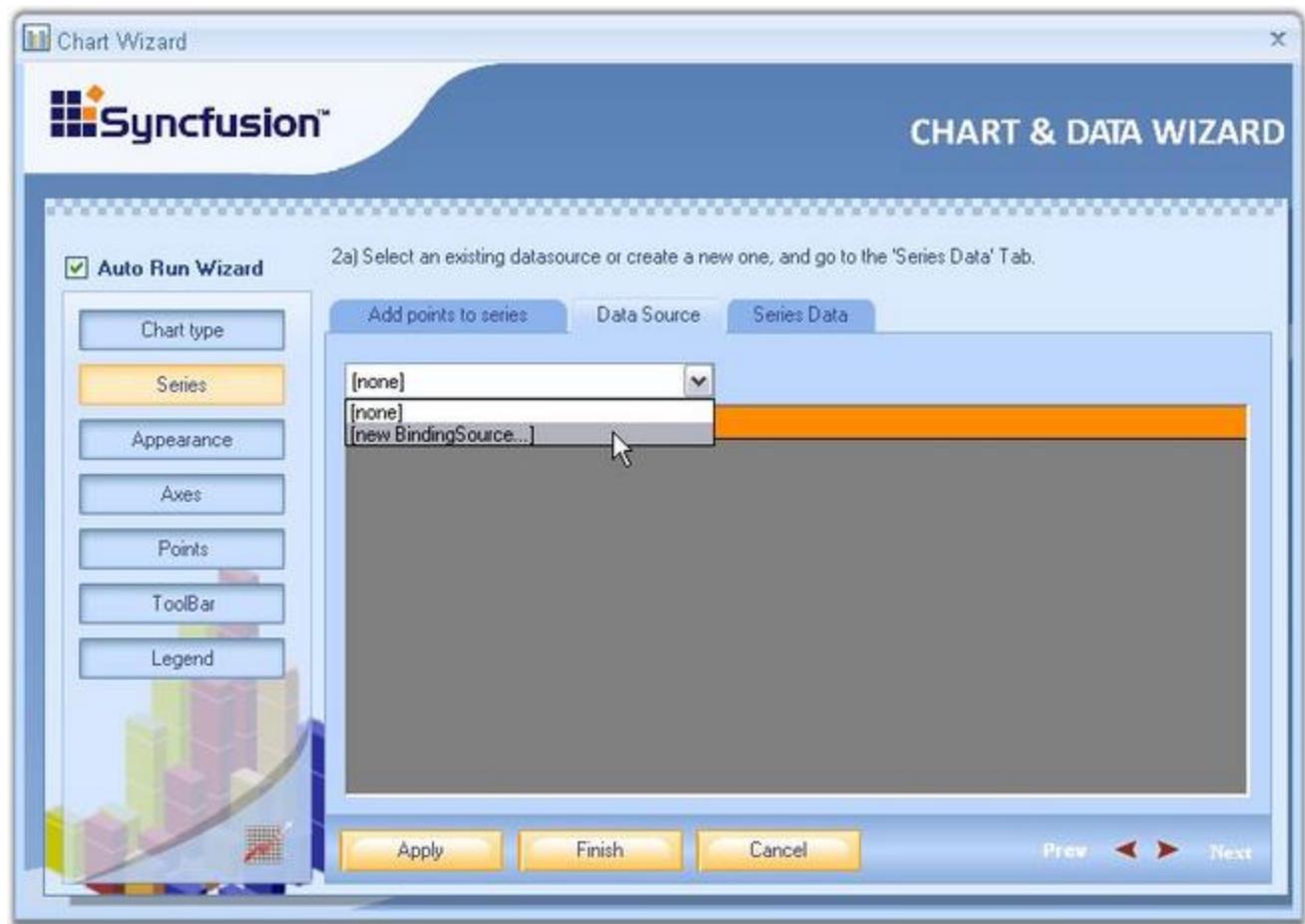
The below steps lets you bind a database table with the ChartControl.

1. Open the **Chart Wizard** tool, Click **Series** button and go to the **Data Source** tab as shown in the image below.



*Figure 28: Data Source Tab*

2. First step is to select the chart data source from the drop-down list. All data sources available with the form will be shown in the list. If there is no data source in the list, click the **new BindingSource** option from the drop-down list.



*Figure 29: Selecting "new BindingSource...." from the drop-down list in the Data Source Tab*

3. This opens a **Data Source Configuration Wizard**. Choose the Data source Type as **Database**, and click **Next**.



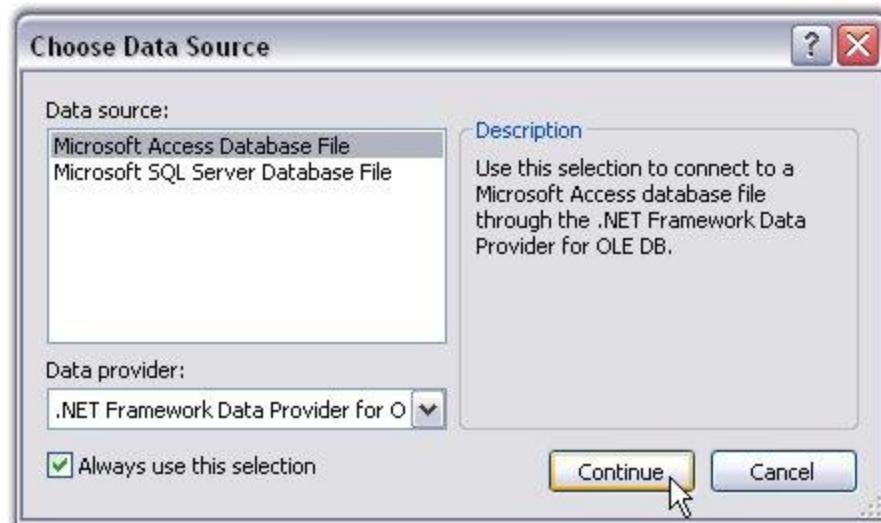
*Figure 30: Selecting the Data Source Type*

4. Then click **New Connection**.



*Figure 31: Creating a New Connection*

5. In the Choose Data Source dialog box, select the data source as MS SQL server database or MS Access database, and then click Continue button.



*Figure 32: Choose Data Source Dialog Box*

6. This opens the **Add Connection** dialog box. Click the **Browse** button and select the database file from any location. Click **OK** to make this connection available to the Data source Configuration Wizard.



*Figure 33: Selecting the Database file by clicking on the Browse Button in the Add Connection Dialog Box*

7. You will be directed to the Data Source Configuration Wizard after completing the above steps. Click **Next**.



*Figure 34: Next button in the Data Source Configuration Wizard is Clicked*

8. Tables and Views that are available in the selected database will be listed in the Wizard. Select the appropriate table, required columns and then click **Finish**.

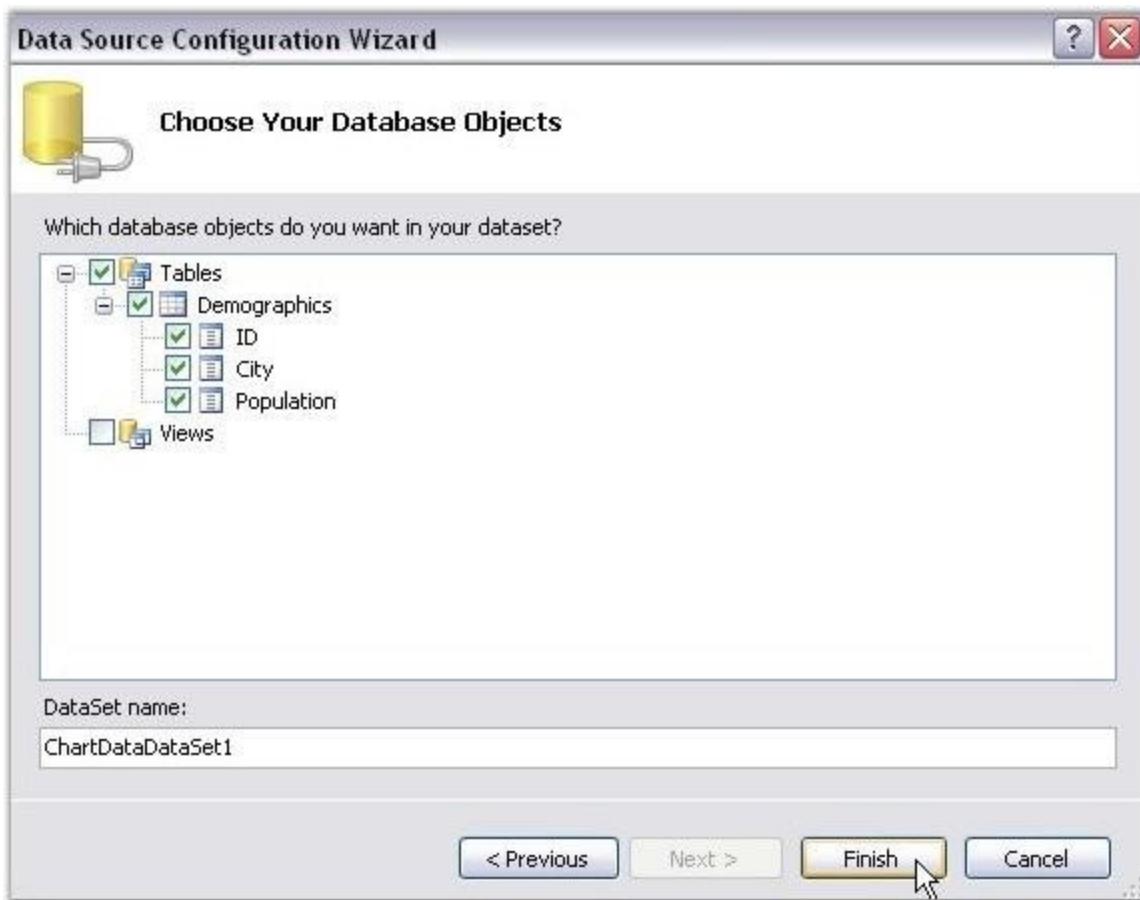
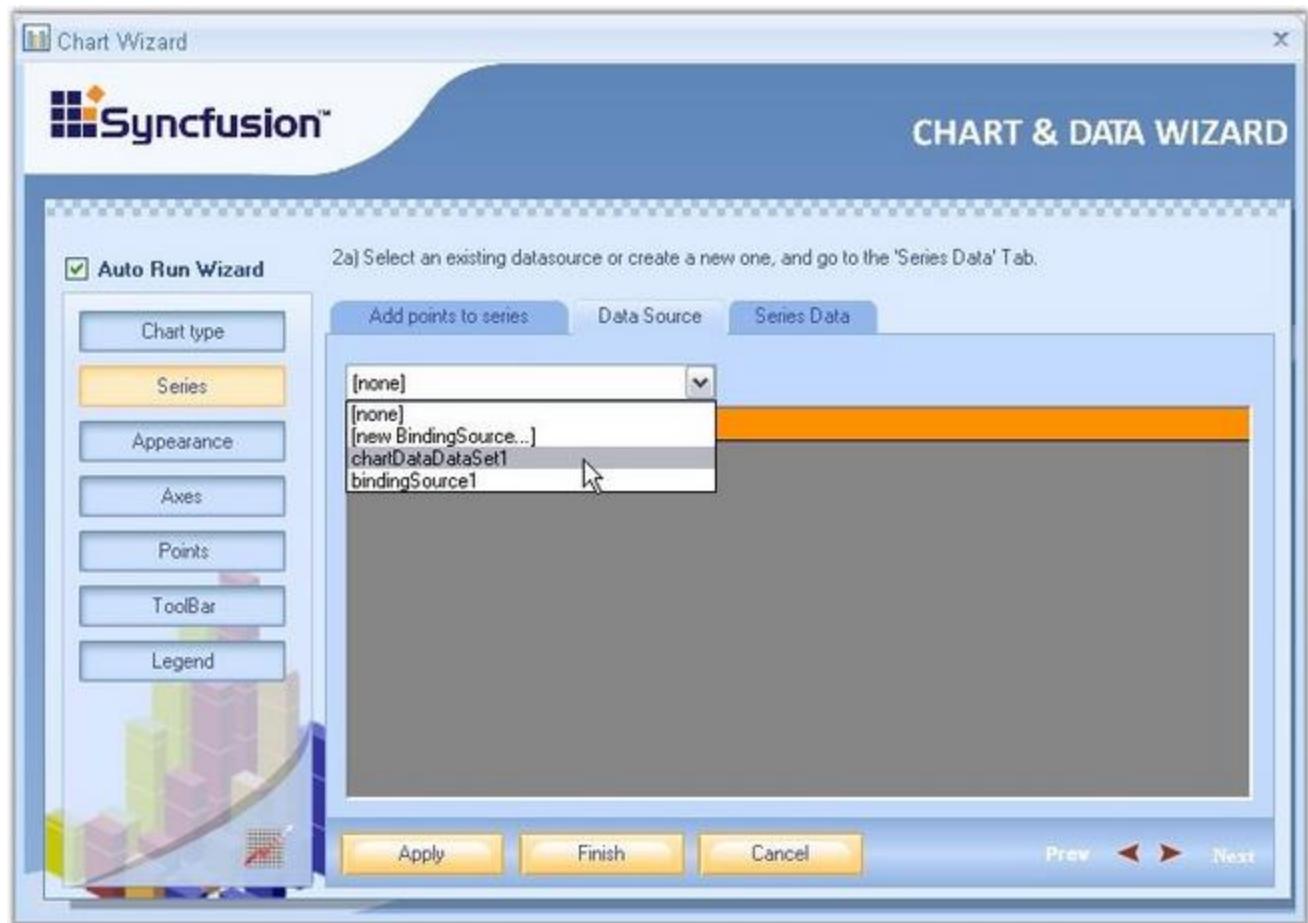


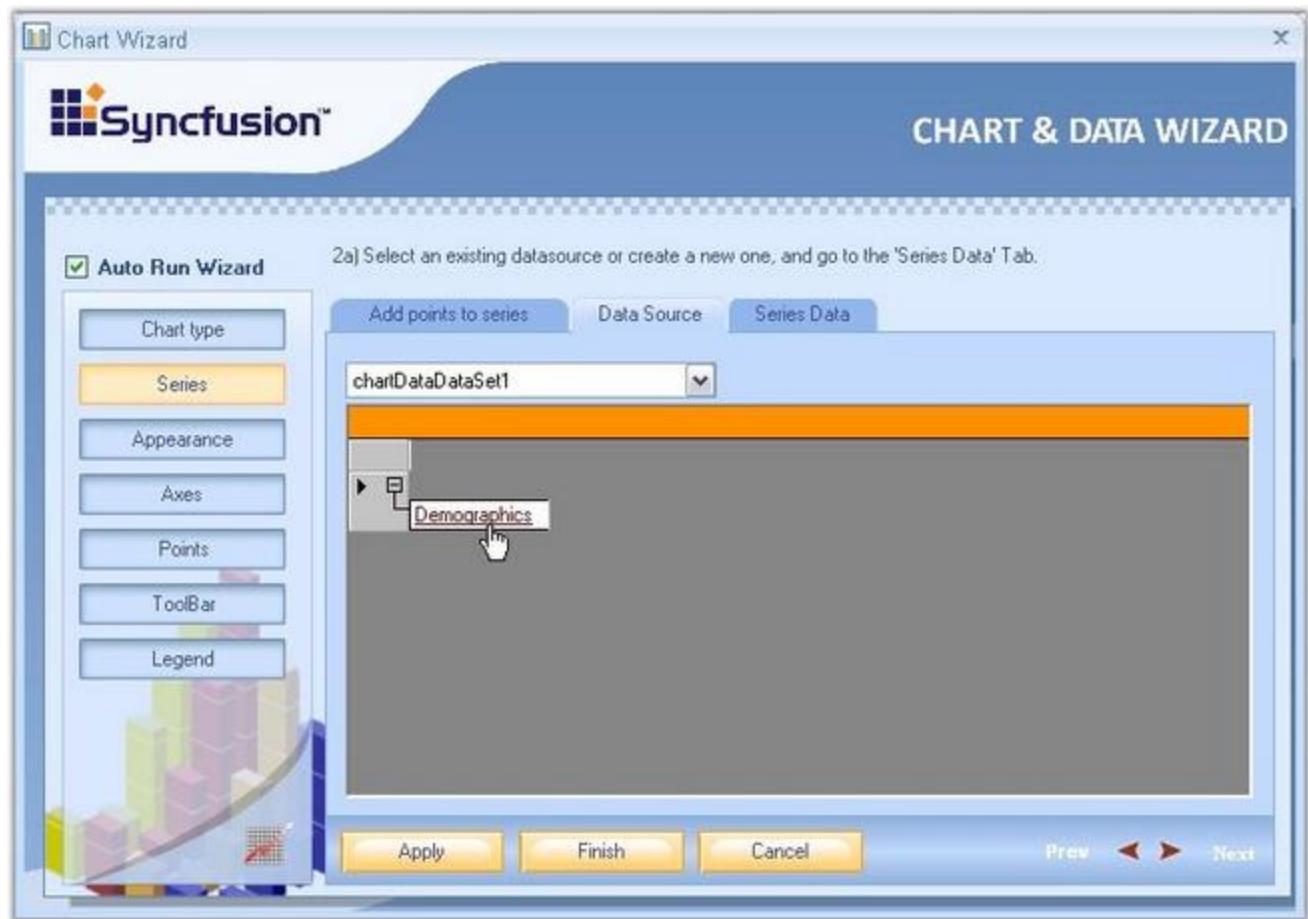
Figure 35: Finish button in the Data Source Configuration Wizard is clicked after selecting the required Table and Columns

9. You will be directed back to the Chart Wizard now. Select the database from the **Data Source** list as shown in the image below.



*Figure 36: Selecting the Database from the drop-down List in the Data Source Tab*

10. Once the source is selected, the selected table will be visible as in the below image.



*Figure 37: Selected Table*

#### **Binding the Table Data with Chart Series**

1. Click the 'Series Data' option in the wizard to select the series to which the data is to be bound. In 'Series Data' page, select the series using the **Series Data** box.



*Figure 38: Selecting the Series to which the data is to be Bound*

2. To assign the retrieved database column to X and Y values of the series, use **X Value** box and **Y Value** box as shown in the below screen shots.

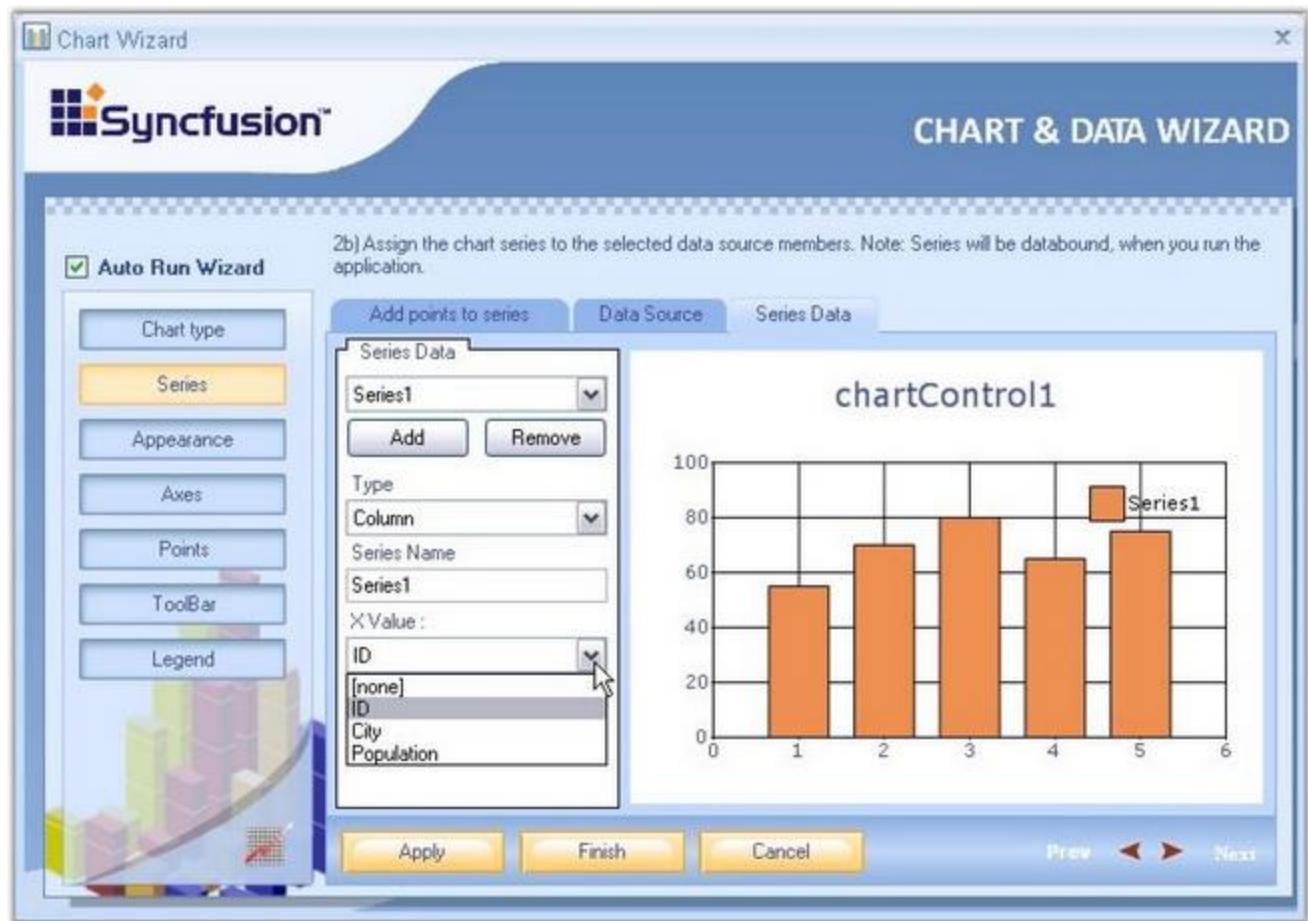
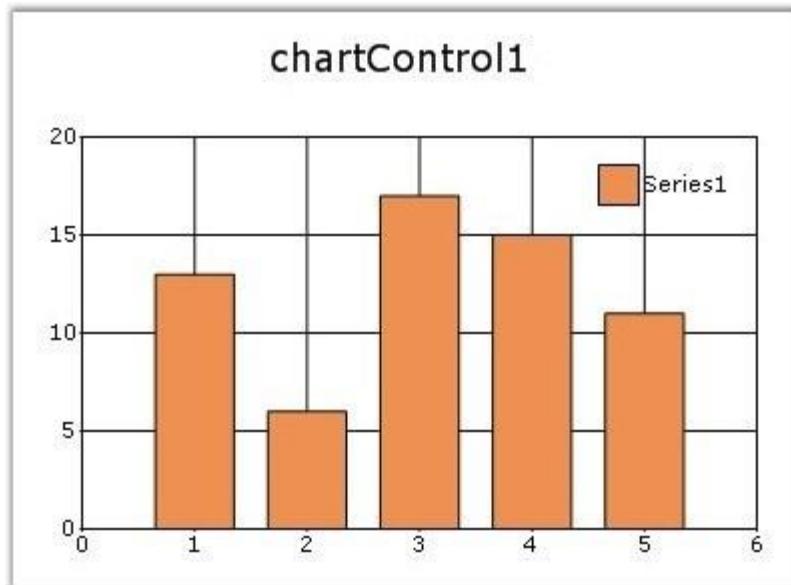


Figure 39: Assigning the retrieved database column to X value of the Series



*Figure 40: Assigning the retrieved database column to Y value of the Series*

3. Click **Finish** to apply these data binding settings to the Chart. The below image illustrates the Chart bound with custom data.



*Figure 41: Data Source bound to the Chart by using the Chart Wizard*

## 4.3 Improving Performance

### Properties and Methods used to improve chart performance

Chart control	Description
Property	
ChartControl.ImprovePerformance	<p>Instructs the chart to calculate the axes ranges only before painting and not at every series adding or moving. Performance of a Chart with large number of series will be improved significantly, if this property is enabled.</p> <p>By default this property is false.</p>

CalcRegions	This property by default is <b>true</b> . This controls the Tooltips, Autohighlighting properties and RegionHit events. If these properties and events are not used, this property can be set to false for better performance.
ChartSeries.EnableStyles	Disabling this property will in turn disable the point symbols and point text which speeds up the chart.
ChartSeries.Style.DisplayShadow	Setting this property to false will not render the series with shadow, which will increase the speed of the chart.  By default this property is set to <b>false</b> .
Indexed	The chart renders faster if the series is not indexed. This of course, may or may not be possible in all cases.  By default this property is <b>false</b> .
BackInterior	The background style for the Chart control is specified using this property and if this property is not set with gradient or pattern style, will help improve the performance of the chart.

ChartArea.BackInterior	This property sets the back color for the chart area. If not set with gradient or pattern style, will help improve the performance of the chart.
ChartInterior	If this property which fills the chart interior, not set with gradient or pattern style, will improve the performance of the chart.
<b>Method</b>	
BeginUpdate and EndUpdate	Encapsulate your "data points adding code" within <b>BeginUpdate</b> and <b>EndUpdate</b> to improve Chart initialization speed. See the example below.

**[C#]**

```
// Improves the performance of the chart when a large number of series
// are used.
this.chartControl1.ImprovePerformance = true;

this.chartControl1.CalcRegions = false;

this.chartControl1.Series[0].EnableStyles = false;

this.chartControl1.Series[0].Style.DisplayShadow = false;

this.chartControl1.Indexed = true;

//BeginUpdate and EndUpdate methods
private DataModel datamodel1;
ChartSeries series = this.chartControl1.Model.NewSeries("Line 1",
ChartSeriesType.Line);

this.chartControl1.BeginUpdate();

// Add a whole bunch of points to the series like this:
series.Points.Add(1, 10), etc.
```

```
this.chartControl1.EndUpdate();
```

**[VB.NET]**

```
' Improves the performance of the chart when a large number of series  
are used.  
Me.chartControl1.ImprovePerformance = True  
  
Me.chartControl1.CalcRegions = False  
  
Me.chartControl1.Series[0].EnableStyles = False  
  
Me.chartControl1.Series[0].Style.DisplayShadow = False  
  
Me.chartControl1.Indexed = True  
  
'BeginUpdate and EndUpdate methods  
Private datamodel1 As DataModel  
Private series As ChartSeries = Me.chartControl1.Model.NewSeries("Line  
1", ChartSeriesType.Line)  
  
Me.chartControl1.BeginUpdate()  
  
'Add a whole bunch of points to the series like this:  
series.Points.Add(1, 10), etc.  
  
Me.chartControl1.EndUpdate()
```

## 4.4 Chart Types

Essential Chart includes a comprehensive set of more than 35 Chart types for all your business needs. Each one is highly and easily configurable with built-in support for creating stunning visual effects.

Chart types are specified on each **ChartSeries** through the **Type** property. All the chart types are required to have at least one **X** and one **Y** value. Certain chart types need more than one **Y** value.

The following table narrates the minimum and maximum number of series and number of Y values required by each type of chart supported by Essential Chart.

Chart Type	Minimum Number of Series	Maximum Number of Series	Number of Y Values Required
<a href="#">Area Charts</a>	1	Unlimited	1
<a href="#">Bar Charts</a>	1	Unlimited	1
<a href="#">Box And Whisker Charts</a>	1	Unlimited	5
<a href="#">Bubble Charts</a>	1	Unlimited	2
<a href="#">Candle Charts</a>	1	Unlimited	4
<a href="#">Column Charts</a>	1	Unlimited	1
<a href="#">Column Range Charts</a>	1	Unlimited	2
<a href="#">Combination Charts</a>	2	Unlimited	1
<a href="#">Funnel Charts</a>	1	1	1
<a href="#">Gantt Charts</a>	1	Unlimited	2
<a href="#">Hi Lo Charts</a>	1	Unlimited	4
<a href="#">Hi Lo Open Close Charts</a>	1	Unlimited	4
<a href="#">Histogram Charts</a>	1	Unlimited	1
<a href="#">Kagi Charts</a>	1	Unlimited	1
<a href="#">Line Charts</a>	1	Unlimited	1
<a href="#">Pie Charts</a>	1	1	1
<a href="#">Point And Figure Charts</a>	1	Unlimited	2
<a href="#">Polar Charts</a>	1	Unlimited	1
<a href="#">Pyramid Charts</a>	1	1	1
<a href="#">Radar Charts</a>	1	Unlimited	1
<a href="#">Renko Charts</a>	1	Unlimited	1
<a href="#">Rotated Spline Charts</a>	1	Unlimited	1

<a href="#">Scatter Charts</a>	1	Unlimited	1
<a href="#">Spline Area Charts</a>	1	Unlimited	1
<a href="#">Spline Charts</a>	1	Unlimited	1
<a href="#">Stacking Area Charts</a>	2	Unlimited	1
<a href="#">Stacking Bar Charts</a>	2	Unlimited	1
<a href="#">Stacking Column Charts</a>	2	Unlimited	1
<a href="#">Step Area Charts</a>	1	Unlimited	1
<a href="#">Step Line Charts</a>	1	Unlimited	1
<a href="#">Three Line Break Charts</a>	1	Unlimited	1
<a href="#">Tornado Charts</a>	1	Unlimited	2

#### 4.4.1 Line Charts

Line charts typically use a line to connect the different data points in a series. Such lines are either straight, splines or steps. Line charts are simpler and hence also let you visualize multiple series without overlapping like in a bar chart.

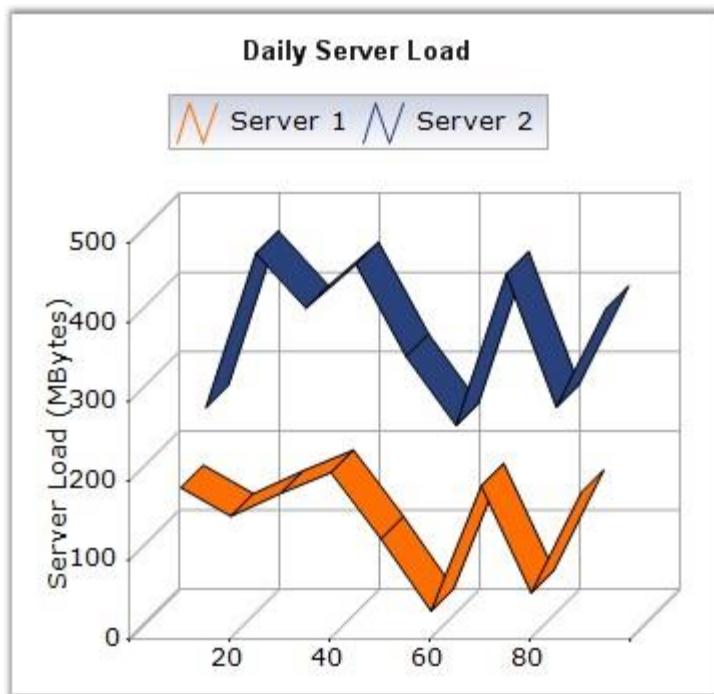
Here are the different types of Line Charts.

##### 4.4.1.1 Line Chart

Line Charts join points on a plot using straight lines showing trends in data at equal intervals. Line charts treat the input as non-numeric, categorical information, equally spaced along the x-axis. This is appropriate for categorical data, such as text labels, but can produce unexpected results when the X values consist of numbers.

When rendered in 3D, the plot looks like a ribbon and hence such types are also referred to as Ribbon or Strip Charts.

The appearance of the lines and the points can be configured with options such as the colors used, thickness of the lines and the symbols displayed.



*Figure 42: Chart displaying Line Series in 3D Mode*

### Chart Details

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar, Polar, Radar.

Line series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.Line);  
series.Points.Add(1, new double[] { 20, 8, 8 });  
series.Points.Add(2, new double[] { 70, 5, 5 });
```

```
series.Points.Add(3, new double[] { 10, 8, 8 });
series.Points.Add(4, new double[] { 40, 10, 10 });

// Add the series to the chart series collection.
this.chartControl1.Series.Add(series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.Line)
series.Points.Add(1, new double[] { 20, 8, 8 })
series.Points.Add(2, new double[] { 70, 5, 5 })
series.Points.Add(3, new double[] { 10, 8, 8 })
series.Points.Add(4, new double[] { 40, 10, 10 })

' Add the series to the chart series collection.
Me.chartControl1.Series.Add(series)
```

#### 4.4.1.2 Spline Chart

Spline Chart is similar to a Line Chart except that it connects the different data points using splines instead of straight lines.

When rendered in 3D, the plot looks like a ribbon and hence such types are also referred to as Ribbon or Strip Charts.

The appearance of the lines and the points can be configured with options such as the colors used, thickness of the lines and the symbols displayed.

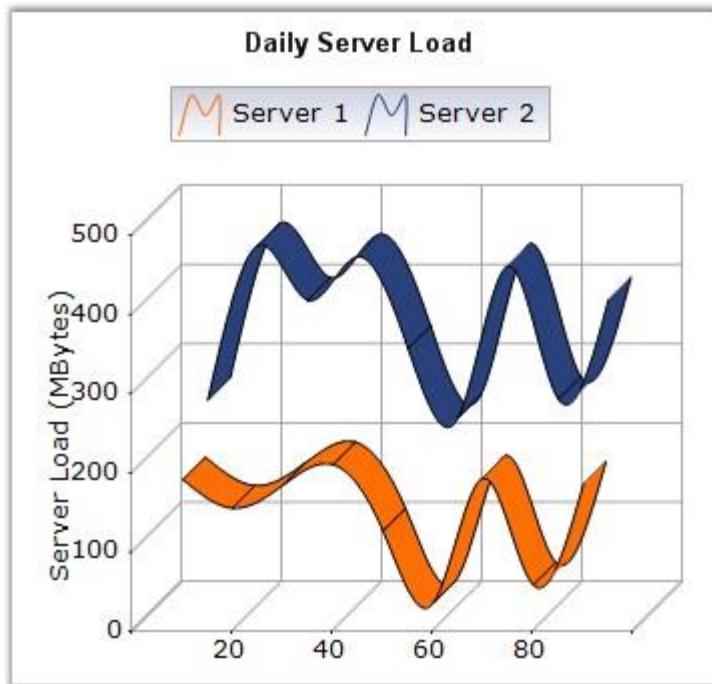


Figure 43: Chart displaying a Spline Series

### Chart Details

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar, Polar, Radar.

Spline series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControll.Model.NewSeries ("Series
Name",ChartSeriesType.Spline);
series.Points.Add(0, 2);
series.Points.Add(1, 3);
series.Points.Add(2, 1);
series.Points.Add(3, 1.5);
series.Points.Add(4, 4);
```

```
series.Points.Add(5, 1);

// Add the series to the chart series collection.
this.chartControll1.Series.Add (series);
```

**[VB .NET]**

```
' Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControll1.Model.NewSeries ("Series
Name", ChartSeriesType.Spline)
series.Points.Add(0, 2)
series.Points.Add(1, 3)
series.Points.Add(2, 1)
series.Points.Add(3, 1.5)
series.Points.Add(4, 4)
series.Points.Add(5, 1)

' Add the series to the chart series collection.
Me.chartControll1.Series.Add (series)
```

**Customization Options**

DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, HighlightInterior,  
ImageIndex, Images, Rotate

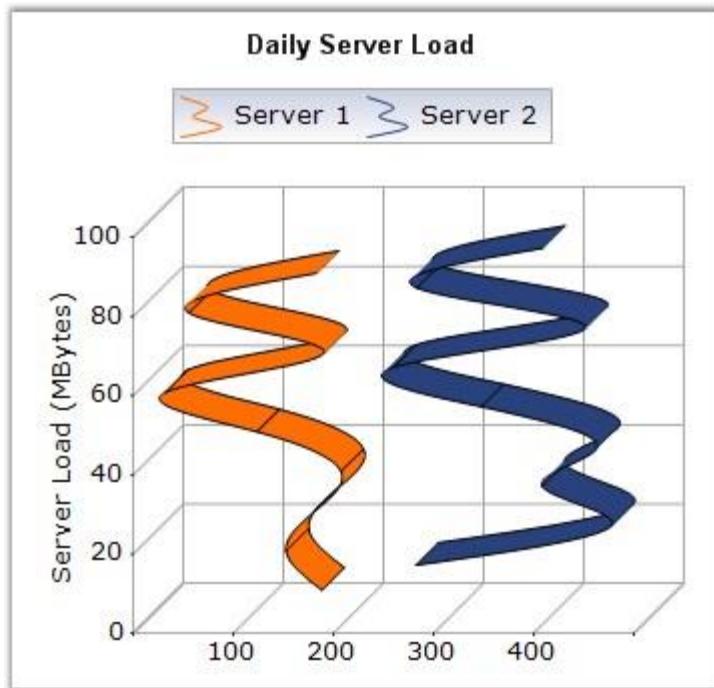
Spacing Between Series, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior,  
LegendItem, Name, PointsToolTipFormat, SmartLabels,

Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.1.3 Rotated Spline Chart

A Rotated Spline Chart is similar to an ordinary Spline Chart. The only difference is that it would be rotated. It plots one or several series of data, and joins each series by smooth, rotated spline curves instead of straight lines.

The following image shows a sample Rotated Spline Chart.



*Figure 44: Chart displaying a Rotated Spline Series*

#### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar, Polar, Radar.

Rotated Spline series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.  
ChartSeries series1 = this.chartControl1.Model.NewSeries(" Series  
1",ChartSeriesType.RotatedSpline );  
  
series1.Points.Add(1, 326);  
series1.Points.Add(2, 491);  
series1.Points.Add(3, 382);  
series1.Points.Add(4, 482);
```

```
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series1);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries(" Series  
1",ChartSeriesType.RotatedSpline )  
  
series1.Points.Add(1, 326)  
series1.Points.Add(2, 491)  
series1.Points.Add(3, 382)  
series1.Points.Add(4, 482)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series1)
```

**Customization Options**

DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, HighlightInterior,  
ImageIndex, Images, Spacing Between Series  
ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels,  
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.1.4 Step Line Chart

Step Line Charts use horizontal and vertical lines to connect data points resulting in a step like progression.

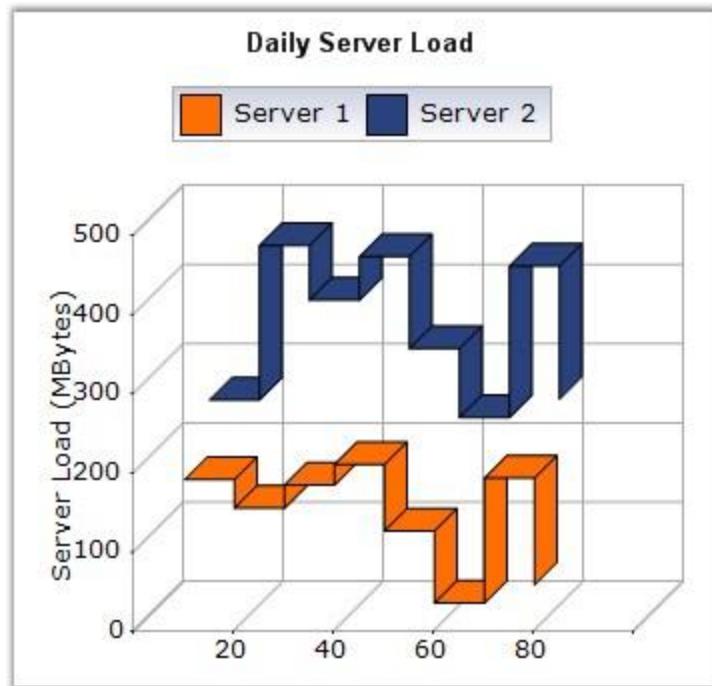


Figure 45: Chart displaying Step Line Series

### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar, Polar, Radar.

Step Line series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControll.Model.NewSeries ("Series
Name",ChartSeriesType.StepLine);
series.Points.Add (0, 1);
series.Points.Add (1, 3);
series.Points.Add (2, 2);
series.Points.Add (3, 2);

// Add the series to the chart series collection.
```

```
this.chartControl1.Series.Add (series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series  
Name", ChartSeriesType.StepLine)  
series.Points.Add (0, 1)  
series.Points.Add (1, 3)  
series.Points.Add (2, 2)  
series.Points.Add (3, 2)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add (series)
```

**Customization Options**

DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, HighlightInterior,  
HitTestRadius, ImageIndex, Images, Rotate  
Spacing Between Series, ShadowInterior, ShadowOffset, StepItem.Inverted, FancyToolTip, Font,  
Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,  
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

## 4.4.2 Bar Charts

Bar Charts are the simplest and most versatile of statistical diagrams. Displayed in horizontal bars, these charts are used to compare values across categories, for showing the variations in the value of an item over time.

A very similar, more common, chart type is the Column Charts where the bars are rendered vertically.

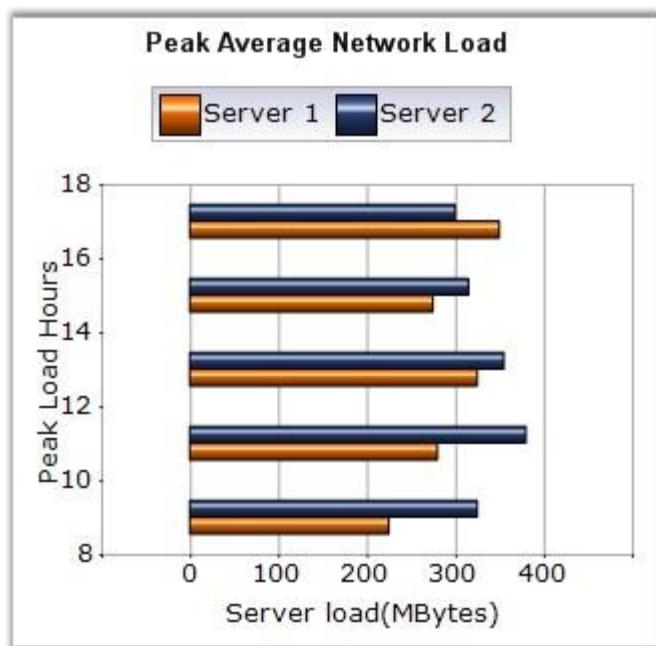
Essential Chart supports these different types of Bar Charts:

### 4.4.2.1 Bar Chart

Bar Chart is the simplest and most versatile of statistical diagrams. It displays horizontal bars for each point in the series and points from adjacent series are drawn as bars next to each other. It is also available with a 3D visual effect. Bar Charts can be used to compare values across categories, for showing the variations in the value of an item over time or for showing the values of several items at a single point in time.

Another good reason to use bar charts is when you realize that the number of a data series fits better in a horizontal format. If you have long gaps between different values and you also have many items to compare, the bar chart type is the best one to use.

The following image shows a multi series Bar Chart.



*Figure 46: Chart displaying Bar Series*

#### **Chart Details**

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Any chart type except Bar and Stacked Bar charts.

Bar series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Bar);  
series.Points.Add(0, 1);  
series.Points.Add(1, 3);  
series.Points.Add(2, 4);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB .NET]**

```
' Create chart series and add data point into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Bar)  
series.Points.Add(0, 1)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

## Customization Options

[Border](#), [ColumnDrawMode](#), [DisplayShadow](#), [DisplayText](#), [DrawSeriesNameInDepth](#), [ElementBorders](#),  
[HighlightInterior](#), [ImageIndex](#), [Images](#)  
[LightAngle](#), [LightColor](#), [PhongAlpha](#), [Rotate](#), [Spacing](#), [Spacing Between Series](#), [ShadingMode](#), [ShadowInterior](#),  
[ShadowOffset](#), [FancyToolTip](#), [Font](#), [Interior](#), [LegendItem](#), [Name](#), [PointsToolTipFormat](#), [SmartLabels](#), [Summary](#),  
[Text](#), [TextColor](#), [TextFormat](#), [TextOffset](#), [TextOrientation](#), [Visible](#)

## See Also

[Column Chart](#)

#### 4.4.2.2 Stacking Bar Chart

Stacking Bar Charts are similar to regular bar charts except that the Y values stack on top of each other in the specified series order. This helps visualizing the relationship of parts to the whole.

The following image shows a sample Stacking Bar Chart.

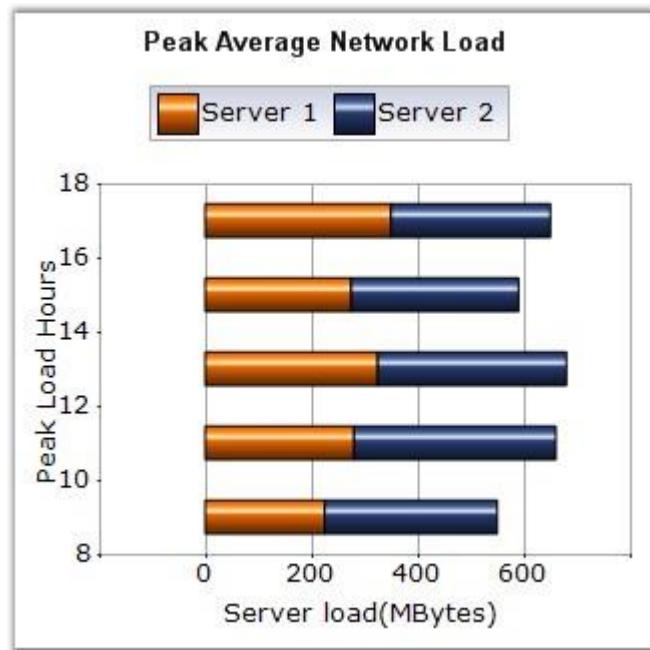


Figure 47: Chart displaying Stacking Bar Series

#### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	Two or More (Single series is rendered just as a bar).
<b>Cannot be Combined with</b>	Any chart type except Bar and Stacked Bar charts.

Stacking bar series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data point into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.StackingBar);  
series.Points.Add(0, 1);  
series.Points.Add(1, 3);  
series.Points.Add(2, 4);  
  
ChartSeries series2 = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.StackingBar);  
series2.Points.Add(0, 2);  
series2.Points.Add(1, 1);  
series2.Points.Add(2, 1);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);  
this.chartControl1.Series.Add(series2);
```

**[VB .NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.StackingBar)  
series.Points.Add(0, 1)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
  
Dim series2 As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.StackingBar)  
series2.Points.Add(0, 2)  
series2.Points.Add(1, 1)  
series2.Points.Add(2, 1)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)  
Me.chartControl1.Series.Add(series2)
```

### Customization Options

Border, ColumnDrawMode, DisplayText, DrawSeriesNameInDepth, ElementBorders,  
HighlightInterior, ImageIndex, Images

LightAngle, LightColor, Rotate, Spacing, Spacing Between Series, ShadingMode,  
ShadowInterior, ShadowOffset, ZOrder, FancyToolTip, Font, Interior, LegendItem, Name,

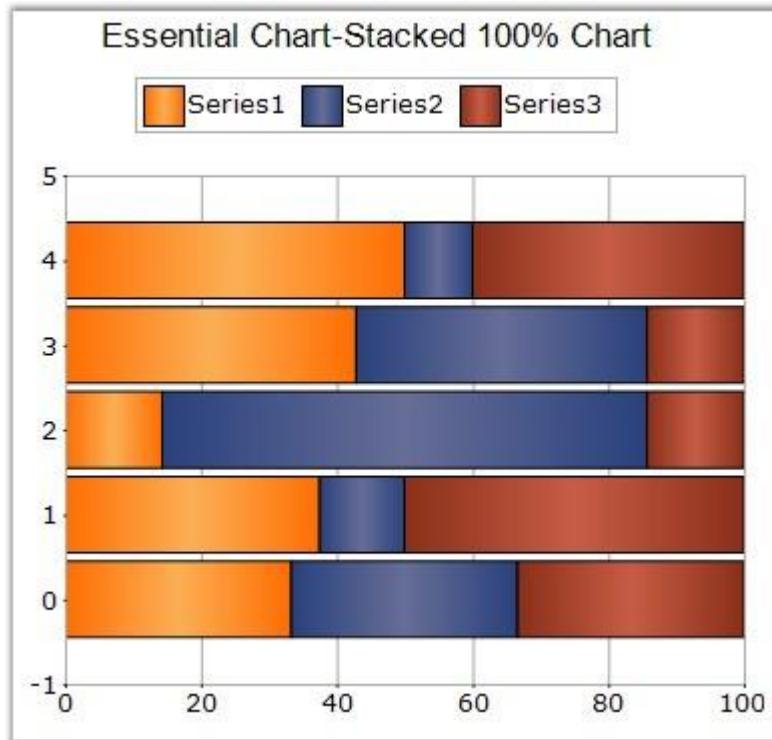
PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

**See Also**

[Stacking Area Chart](#), [Stacking Column Chart](#)

#### 4.4.2.3 StackedBar100 Chart

This chart type displays multiple series of data as stacked Bars ensuring that the cumulative proportion of each stacked element always totals 100%. The y-axis will hence always be rendered with the range 0 - 100.



*Figure 48: A 100% StackedBar Chart*

**Details**

<b>Number of Y values per point</b>	1
<b>Number of Series</b>	Two or more.
<b>SupportMarker</b>	No
<b>Cannot be Combined with</b>	Any other chart types.

**[C#]**

```
ChartSeries series1 = this.chartControl1.Model.NewSeries("chart",
ChartSeriesType.StackingBar100);
series1.Points.Add(0, 25.3);
series1.Points.Add(1, 45.7);
series1.Points.Add(2, 97.3);
series1.Points.Add(3, 20.6);
series1.Points.Add(4, 125.8);
series1.Points.Add(5, 216.1);
this.chartControl1.Series.Add(series1);
```

**[VB .NET]**

```
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("chart",
ChartSeriesType.StackingBar100)
series1.Points.Add(0,25.3)
series1.Points.Add(1,45.7)
series1.Points.Add(2,97.3)
series1.Points.Add(3,20.6)
series1.Points.Add(4,125.8)
series1.Points.Add(5,216.1)
Me.chartControl1.Series.Add(series1)
```

### **Customization Options**

Border, ColumnDrawMode, DisplayText, DrawSeriesNameInDepth, ElementBorders, HighlightInterior, ImageIndex, Images

LightAngle, LightColor, Rotate, Spacing, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset, ZOrder, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

### **See Also**

[StackedArea100 Chart](#), [StackedColumn100 Chart](#)

#### 4.4.2.4 Gantt Chart

A Gantt chart is a graphical representation of the duration of tasks against the progression of time. In a Gantt chart, each task takes up one row. The expected time for each task is represented by a horizontal bar whose left end marks the expected beginning of the task and whose right end marks the expected completion of the task. Tasks may run sequentially, in parallel or overlapping.

You could then use another series to represent the completed portion of the different tasks. This new series will then contain data points with their beginning values coinciding with the beginning values of the data points from the previous series and the ending value based on the fraction of the work that has been completed on the task. This way, one can get a quick reading of a project progress by drawing a vertical line through the chart at the current date.

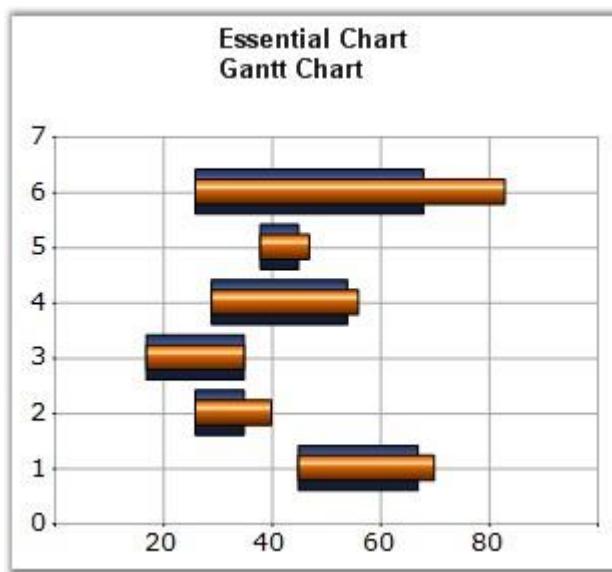


Figure 49: Chart displaying Gantt Series

#### Chart Details

Details	
Number of Y values per point	2. (1st is beginning value and the 2nd is the ending value)
Number of Series	One or more.

<b>Cannot be Combined with</b>	Pie, Bar, Polar, Radar.
--------------------------------	-------------------------

Gantt series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Gantt);  
series.Points.Add(0, 1, 5);  
series.Points.Add(1, 3, 7);  
series.Points.Add(2, 4, 8);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

[C#]

```
// Create chart series and add data points into it.  
Dim series As ChartSeries = Me.ChartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Gantt)  
series.Points.Add(0, 1, 5)  
series.Points.Add(1, 3, 7)  
series.Points.Add(2, 4, 8)  
Me.ChartControl1.Series.Add(series)
```

### Customization Options

[Border](#), [ColumnDrawMode](#), [DisplayShadow](#), [DisplayText](#), [ElementBorders](#), [GanttDrawMode](#), [HighlightInterior](#), [ImageIndex](#), [Images](#)

[LightAngle](#), [LightColor](#), [PhongAlpha](#), [PointWidth](#), [RelatedPoints](#), [Spacing](#), [Spacing Between Series](#), [ShadingMode](#), [ShadowInterior](#), [ShadowOffset](#)

[ZOrder](#), [FancyToolTip](#), [Font](#), [Interior](#), [LegendItem](#), [Name](#), [PointsToolTipFormat](#), [SmartLabels](#), [Summary](#), [Text](#), [TextColor](#), [TextFormat](#), [TextOffset](#), [TextOrientation](#), [Visible](#)

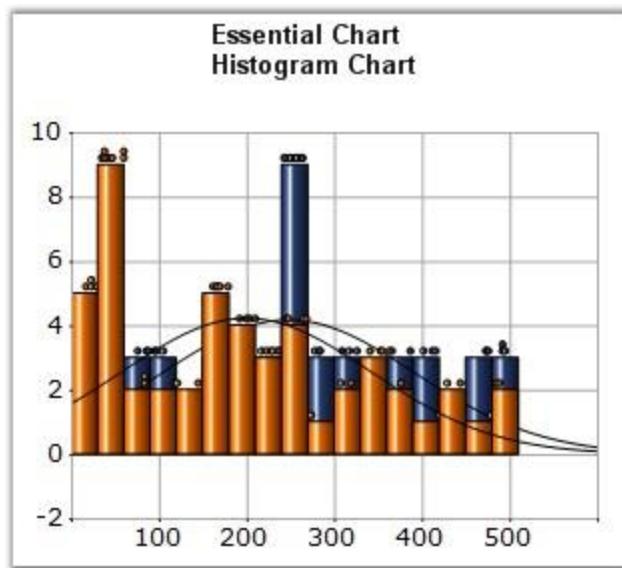
## 4.4.2.5 Histogram Chart

Histogram is a bar (column) chart of a frequency distribution in which the widths of the bars are proportional to the classes into which the variable has been divided, and the heights of the bars are proportional to the class frequencies. The categories are usually specified as non overlapping intervals of some variable. The categories (bars) must be adjacent. In addition, the chart has the capability to draw a normal distribution curve.

Histograms are useful data summaries that convey the following information:

- The general shape of the frequency distribution. (normal, exponential, etc.)
- Symmetry of the distribution and whether it is skewed.
- Modality - unimodal, bimodal or multimodal.

The shape of the distribution conveys important information such as the probability distribution of the data.



*Figure 50: Chart displaying a Histogram Series*

#### **Chart Details**

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One or more.
<b>Cannot be Combined with</b>	Pie, Bar, Polar, Radar.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControll1.Model.NewSeries("System  
1",ChartSeriesType.Histogram);  
series.Text = series.Name;  
  
series.Points.Add( 3, 1020 );  
series.Points.Add( 45, 440 );  
series.Points.Add( 23, 605 );  
. . .  
. . .  
. . .  
. . .  
  
// Add the series to the chart series collection.  
this.chartControll1.Series.Add(series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControll1.Model.NewSeries("System  
1",ChartSeriesType.Histogram)  
series.Text = series.Name  
  
series.Points.Add(3, 1000)  
series.Points.Add(45, 1000)  
series.Points.Add(23, 1000)  
. . .  
. . .  
. . .  
. . .  
. . .  
  
' Add the series to the chart series collection.  
Me.chartControll1.Series.Add(series)
```

**Customization Options**

Border, DisplayShadow, DisplayText, DrawHistogramNormalDistribution, LightAngle, LightColor,  
NumberOfHistogramIntervals, , PhongAlpha

Rotate, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset,  
ShowHistogramDataPoints, FancyToolTip, Font, Interior, LegendItem, Name,

PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.2.6 Tornado Chart

The Tornado chart is a bar chart which shows the variability of an output to several different inputs. Variability is displayed using relative lengths of bars across a range. It is mainly used in sensitivity analysis. It shows how different random factors can influence the prognostic outcome.

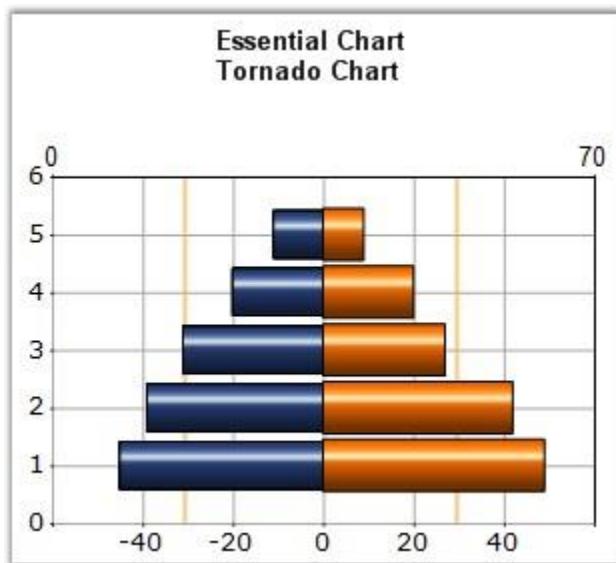


Figure 51: Chart displaying a Tornado Chart

##### Chart Details

Details	
Number of Y values per point	2
Number of Series	One or more
Cannot be Combined with	Pie, Bar, Polar, Radar

The Tornado series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControll1.Model.NewSeries("Series 0",
```

```
ChartSeriesType.Tornado);

series.Points.Add(1, 0, 48);
series.Points.Add(2, 0, 42);
series.Points.Add(3, 0, 27);
series.Points.Add(4, 0, 20);
series.Points.Add(5, 0, 9);

// Add the series to the chart series collection.
this.chartControl1.Series.Add(series);

ChartSeries series2 = this.chartControl1.Model.NewSeries("Series 1",
ChartSeriesType.Tornado);

series2.Points.Add(1, 0, -45);
series2.Points.Add(2, 0, -39);
series2.Points.Add(3, 0, -31);
series2.Points.Add(4, 0, -20);
series2.Points.Add(5, 0, -11);

this.chartControl1.Series.Add(series2);
```

**[VB.NET]**

```
' Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series
0",ChartSeriesType.Tornado)

series.Points.Add(1,0, 48)
series.Points.Add(2, 0, 42)
series.Points.Add(3, 0, 27)
series.Points.Add(4, 0, 20)
series.Points.Add(5, 0, 9)

' Add the series to the chart series collection.
Me.chartControl1.Series.Add(series)

' Create chart series and add data points into it.
Dim series2 As ChartSeries = Me.chartControl1.Model.NewSeries("Series
0",ChartSeriesType.Tornado)

series2.Points.Add(1,0, -45)
series2.Points.Add(2, 0, -39)
series2.Points.Add(3, 0, -31)
series2.Points.Add(4, 0, -20)
series2.Points.Add(5, 0, -11)
```

```
Me.chartControll.Series.Add(series2)
```

### **Customization Options**

Border, DisplayShadow, DisplayText, ElementBorders, HighlightInterior, ImageIndex, Images, LightAngle, LightColor, PhongAlpha  
Spacing, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

## **4.4.3 Column Charts**

Column Charts are among the most commonly used chart types. Displayed in vertical bars (called columns), they depict the different values of one or more items. Points from adjacent series are drawn as bars next to each other. They are ideal for showing the variations in the value of an item over time.

A very similar bar type is the [Bar Charts](#) type where the bars are rendered horizontal.

Essential Chart supports these different Column Charts.

### **4.4.3.1 Column Chart**

Column Charts are among the most common chart types that are being used. It uses vertical bars (called columns) to display different values of one or more items. It is similar to a bar chart except that, here the bars are vertical and not horizontal. Points from adjacent series are drawn as bars next to each other.

It is used for comparing the frequency, count, total or average of data in different categories. It is ideal for showing the variations in the value of an item over time.

The following image shows a multi series Column Chart.



*Figure 52: Chart displaying Column Series*

#### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar.

Column series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.Column);
series.Points.Add(0, 1);
series.Points.Add(1, 3);
series.Points.Add(2, 4);
```

```
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Column)  
series.Points.Add(0, 1)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

**Customization Options**

Border, ColumnDrawMode, ColumnWidthMode, ColumnFixedWidth, DisplayShadow, DisplayText, DrawColumnSeparatingLines, ColumnType, DrawErrorBars  
DrawSeriesNameInDepth, ElementBorders, ErrorBarsSymbolShape, HighlightInterior, ImageIndex, Images, LightAngle, LightColor, PhongAlpha, Rotate  
Spacing, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,  
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.3.2 Column Range Chart

Column Range Chart is similar to the Column Chart except that each column is rendered over a range. Therefore the user must specify the y-axis Starting and Ending values for each point.

The following figure shows a Column Range Chart.



Figure 53: Chart displaying Column Range Series

**Chart Details**

Details	
<b>Number of Y values per point</b>	2
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar.

The following code snippet illustrates this.

**[C#]**

```
// Create chart series and add data points into it.
ChartSeries series = this.ChartControl1.Model.NewSeries("Series 0",ChartSeriesType.ColumnRange);
series.Points.Add(0,100,400);
series.Points.Add(2,300,600);
series.Points.Add(4,200,700);
series.Text = series.Name;

// Add the series to the chart series collection.
this.ChartControl1.Series.Add(series);
```

**[VB .NET]**

```
' Create chart series and add data point into it.
```

```
Dim series As ChartSeries = Me.ChartControl1.Model.NewSeries("Series 0",ChartSeriesType.ColumnRange)
series.Points.Add(0,100,400)
series.Points.Add(2,300,600)
series.Points.Add(4,200,700)
series.Text = series.Name

' Add the series to the chart series collection.
Me.ChartControl1.Series.Add(series)
```

#### **Customization Options**

Border, ColumnDrawMode, ColumnWidthMode, ColumnFixedWidth, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images  
LightAngle, LightColor, Rotate, Spacing, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### **4.4.3.3 Stacking Column Chart**

Stacking Column Charts are similar to regular column charts except that the y values stack on top of each other in the specified series order. This helps visualize the relationship of parts to the whole.

The following image shows a sample Stacking Column Chart.



Figure 54: Chart displaying Stacking Column Series

### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	Two or more (A single series will render just like a bar chart).
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar.

Stacking column series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.StackingColumn);
series.Points.Add(0, 1);
series.Points.Add(1, 3);
series.Points.Add(2, 4);

ChartSeries series2 = this.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.StackingColumn);
series2.Points.Add(0, 2);
series2.Points.Add(1, 1);
```

```
series2.Points.Add(2, 1);

// Add the series to the chart series collection.
this.chartControl1.Series.Add(series);
this.chartControl1.Series.Add(series2);
```

**[VB .NET]**

```
'Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series
Name", ChartSeriesType.StackingColumn)
series.Points.Add(0, 1)
series.Points.Add(1, 3)
series.Points.Add(2, 4)

Dim series2 As ChartSeries = Me.chartControl1.Model.NewSeries("Series
Name", ChartSeriesType.StackingColumn)
series2.Points.Add(0, 2)
series2.Points.Add(1, 1)
series2.Points.Add(2, 1)

'Add the series to the chart series collection.
Me.chartControl1.Series.Add(series)
Me.chartControl1.Series.Add(series2)
```

### Customization Options

Border, ColumnWidthMode, ColumnFixedWidth, DisplayShadow, DisplayText,  
DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images  
LightAngle, LightColor, Rotate, Spacing, Spacing Between Series, ShadingMode,  
ShadowInterior, ShadowOffset, ZOrder, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset,  
TextOrientation, Visible

### See Also

[Stacking Area Chart](#), [Stacking Bar Chart](#)

### 4.4.3.4 Stacked Column100 Chart

This chart type displays multiple series of data as stacked Columns ensuring that the cumulative proportion of each stacked element always totals 100 percent. The y-axis will hence always be rendered with the range 0 - 100.

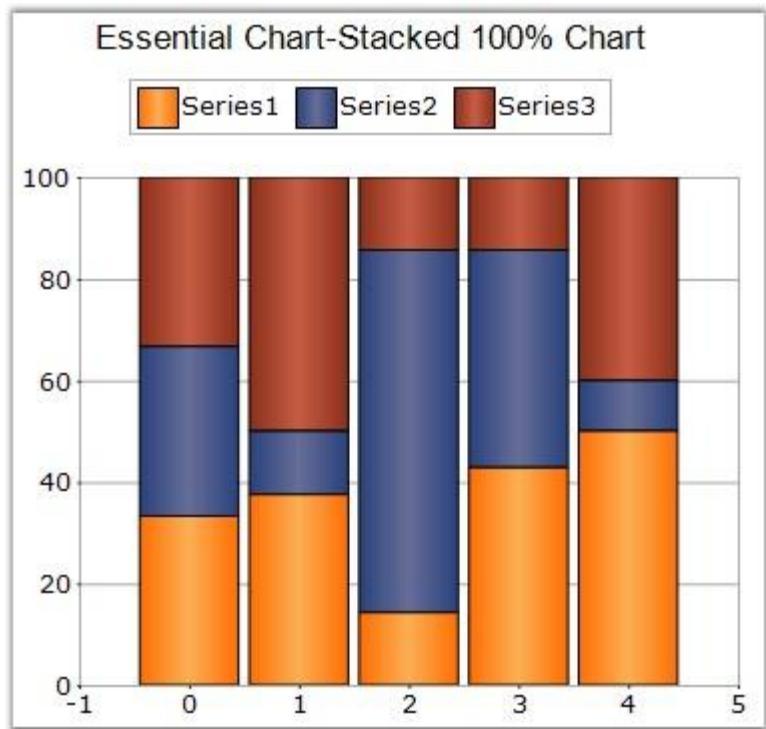


Figure 55: A 100 percent StackedColumn Chart

Details	
Number of Y values per point	1.
Number of Series	Two or more.
SupportMarker	No.
Cannot be Combined with	Doughnut, Pie, Bar, Stacked Bar charts, Polar, Radar, Pyramid, or Funnel.

**[C#]**

```
ChartSeries series1 = this.chartControl1.Model.NewSeries("Series 1",
ChartSeriesType.StackingColumn100);
series1.Points.Add(0, 25.3);
series1.Points.Add(1, 45.7);
series1.Points.Add(2, 97.3);
series1.Points.Add(3, 20.6);
series1.Points.Add(4, 125.8);
series1.Points.Add(5, 216.1);
```

```
this.chartControl1.Series.Add(series1);
```

**[VB .NET]**

```
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("Series", ChartSeriesType.StackingColumn100)
series1.Points.Add(0,25.3)
series1.Points.Add(1,45.7)
series1.Points.Add(2,97.3)
series1.Points.Add(3,20.6)
series1.Points.Add(4,125.8)
series1.Points.Add(5,216.1)
Me.chartControl1.Series.Add(series1)
```

**Customization Options**

Border, ColumnWidthMode, ColumnFixedWidth, DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images

LightAngle, LightColor, Rotate, Spacing, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset, ZOrder, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

**See Also**

[StackedArea100 Chart](#), [StackedBar100 Chart](#)

#### 4.4.4 Area Charts

Area Charts emphasize the degree of change of the values over a period of time. Instead of rendering data as discreet bars or columns, an Area Chart renders them in a continuous ebb and flow pattern as defined against the y-axis.

There is support for alpha-blending multiple series areas. The look and feel is also easily customizable by the user.

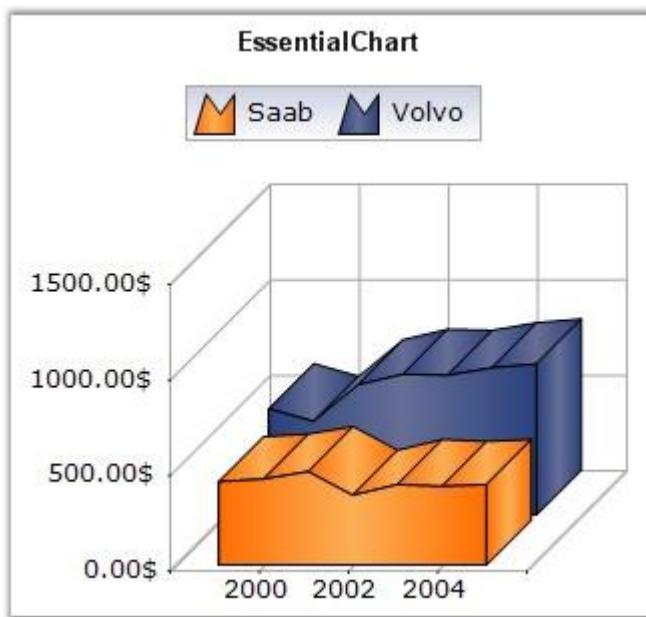
Essential Chart supports these various types of Area Charts:

#### 4.4.4.1 Area Chart

The Area Chart connects the y points using straight lines and forms an area covered by the above lines and x-axis. This area is then shaded with a specified color or gradient.

Multiple series can be plotted on the same chart and alpha-blended interior color can be used on the exterior chart to make the interior chart show through.

The following image shows a multi series Area Chart.



*Figure 56: Chart displaying Area Series in 3D View*

#### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Polar, Radar, Gantt, Stacked Bar

An Area series can be added to the chart using the following code.

**[C#]**

```
// Create a chart series and add data points into it.  
ChartSeries series = this.chartControll.Model.NewSeries("Series  
Name",ChartSeriesType.Area);  
series.Points.Add(0, 4.5);  
series.Points.Add(1, 3);  
series.Points.Add(2, 4);  
series.Points.Add(3, 3);  
  
// Add the series to the chart series collection.  
this.chartControll.Series.Add(series);
```

**[VB .NET]**

```
' Create a chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControll.Model.NewSeries("Series  
Name", ChartSeriesType.Area)  
ChartSeries series = this.chartControll.Model.NewSeries("Series  
Name",ChartSeriesType.Area);  
series.Points.Add(0, 4.5)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
series.Points.Add(3, 3)  
  
' Add the series to the chart series collection.  
Me.chartControll.Series.Add(series)
```

**Customization Options**

Border, DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders,  
HighlightInterior, ImageIndex, Rotate, SeriesToolTipFormat

Spacing Between Series, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat,  
SmartLabels,

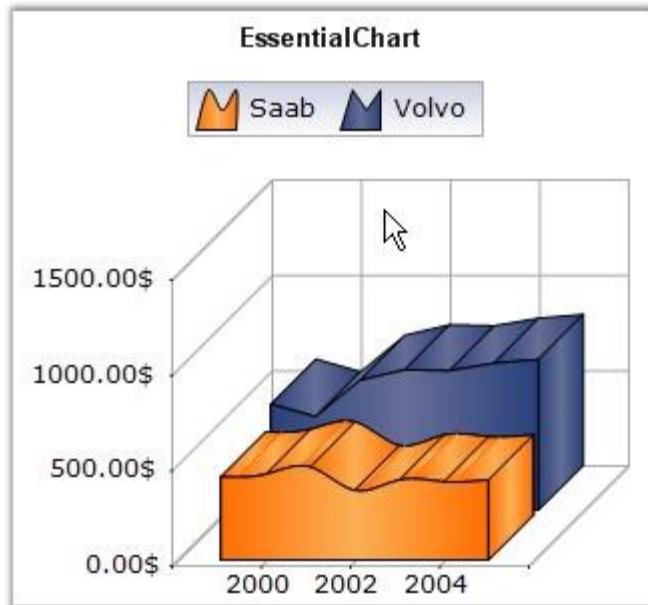
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.4.2 Spline Area Chart

Spline Area Chart is similar to an Area Chart with the only difference being the way in which the points of a series are connected. It connects each series of points by a smooth spline curve. The area enclosed by the chart is filled with specified interior brush.

Multiple series can be plotted on the same chart and alpha-blended interior color can be used on the exterior chart to make the interior chart(s) show through.

The following image shows a multi series Spline Area Chart.



*Figure 57: Chart displaying Spline Area Series in 3D Mode*

#### **Chart Details**

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Polar, Radar and Stacked Bar.



Spline area series can be added to the chart using the following code.

**[C#]**

```
// Create a chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.SplineArea);  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.Area);  
series.Points.Add(0, 4.5);  
series.Points.Add(1, 3);  
series.Points.Add(2, 4);  
series.Points.Add(3, 3);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB .NET]**

```
' Create a chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.SplineArea)  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.Area);  
series.Points.Add(0, 4.5)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
series.Points.Add(3, 3)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

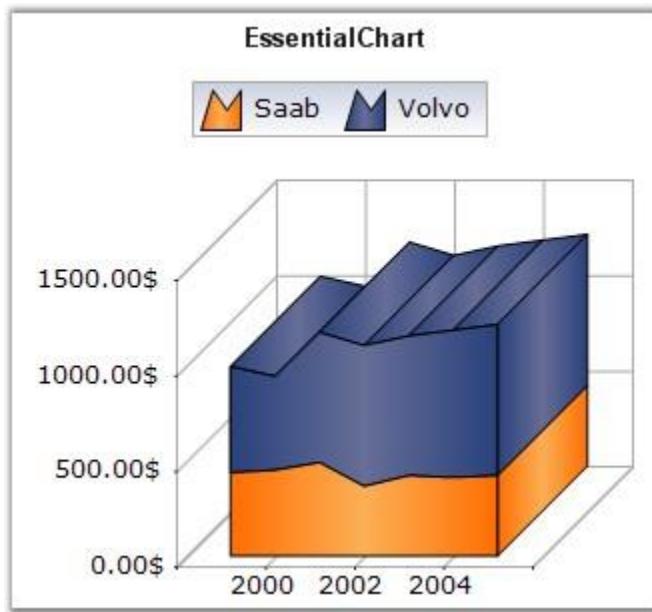
**Customization Options**

Border, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Rotate,  
SeriesToolTipFormat, Spacing Between Series, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset,  
TextOrientation, Visible

#### 4.4.4.3 Stacking Area Chart

Stacking Area Charts are similar to regular area charts except that the y values stack on top of each other in the specified series order. This helps visualize the relationship of parts to the whole.

The following image shows a sample Stacking Area Chart.



*Figure 58: Chart displaying Stacking Area Series in 3D*

#### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar and Radar.

Stacking area series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.StackingArea);  
series.Points.Add(0, 1);  
series.Points.Add(1, 3);  
series.Points.Add(2, 4);  
  
ChartSeries series2 = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.StackingArea);  
series2.Points.Add(0, 2);  
series2.Points.Add(1, 1);  
series2.Points.Add(2, 1);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);  
this.chartControl1.Series.Add(series2);
```

**[VB .NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.StackingArea)  
series.Points.Add(0, 1)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
  
Dim series2 As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.StackingArea)  
series2.Points.Add(0, 2)  
series2.Points.Add(1, 1)  
series2.Points.Add(2, 1)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)  
Me.chartControl1.Series.Add(series2)
```

### Customization Options

Border, DisplayText, DrawSeriesNameInDepth, ElementBorders, HighlightInterior, ImageIndex,  
Rotate, SeriesToolTipFormat, Spacing Between Series  
, ZOrder, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### See Also

[Stacking Bar Chart](#), [Stacking Column Chart](#)

#### 4.4.4.4 StackedArea100 Chart

This chart type displays multiple series of data as stacked areas ensuring that the cumulative proportion of each stacked element always totals 100 percent. The y axis will hence always be rendered with the range 0 - 100.

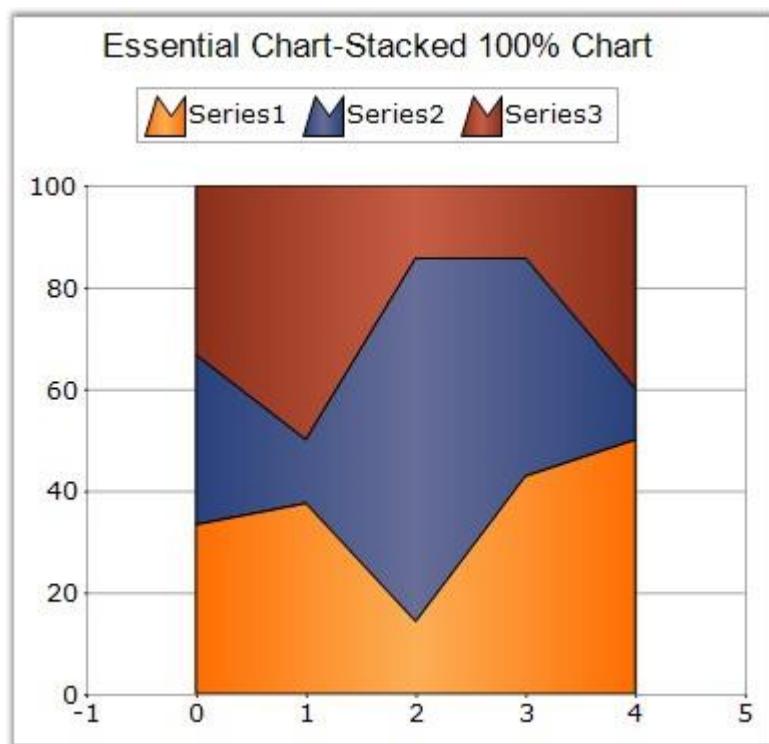


Figure 59: A 100 percent StackedArea Chart

Details	
Number of Y values per point	1
Number of Series	One.

<b>SupportMarker</b>	No
<b>Cannot be Combined with</b>	Any other chart types.

**[C#]**

```
ChartSeries
series1=chartControl1.Model.NewSeries("Series1",ChartSeriesType.StackingArea100);
series1.Points.Add(1,20);
series1.Points.Add(2,30);
series1.Points.Add(3,10);
series1.Points.Add(4,15);
series1.Points.Add(5,25);
this.chartControl1.Series.Add(series1);

ChartSeries
series2=chartControl1.Model.NewSeries("Series2",ChartSeriesType.StackingArea100);
series2.Points.Add(1,20);
series2.Points.Add(2,10);
series2.Points.Add(3,50);
series2.Points.Add(4,15);
series2.Points.Add(5,5);
this.chartControl1.Series.Add(series2);

ChartSeries
series3=chartControl1.Model.NewSeries("Series3",ChartSeriesType.StackingArea100);
series3.Points.Add(1,20);
series3.Points.Add(2,40);
series3.Points.Add(3,10);
series3.Points.Add(4,5);
series3.Points.Add(5,20);
this.chartControl1.Series.Add(series3);
```

**[VB.NET]**

```
Dim series1 As ChartSeries = chartControl1.Model.NewSeries("Series1",
ChartSeriesType.StackingArea100)
series1.Points.Add(0, 20)
series1.Points.Add(1, 30)
series1.Points.Add(2, 10)
series1.Points.Add(3, 15)
series1.Points.Add(4, 25)
Me.chartControl1.Series.Add(series1)
```

```
Dim series2 As ChartSeries = chartControl1.Model.NewSeries("Series2",
ChartSeriesType.StackingArea100)
series2.Points.Add(0, 20)
series2.Points.Add(1, 10)
series2.Points.Add(2, 50)
series2.Points.Add(3, 15)
series2.Points.Add(4, 5)
Me.chartControl1.Series.Add(series2)

Dim series3 As ChartSeries = chartControl1.Model.NewSeries("Series3",
ChartSeriesType.StackingArea100)
series3.Points.Add(0, 20)
series3.Points.Add(1, 40)
series3.Points.Add(2, 10)
series3.Points.Add(3, 5)
series3.Points.Add(4, 20)
Me.chartControl1.Series.Add(series3)
```

### **Customization Options**

Border, DisplayText, DrawSeriesNameInDepth, ElementBorders, HighlightInterior, ImageIndex, Rotate, SeriesToolTipFormat, Spacing Between Series  
, ZOrder, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

### **See Also**

[StackedBar100 Chart](#), [StackedColumn100Chart](#)

#### **4.4.4.5 Step Area Chart**

Step Area Charts are similar to regular area chart except that instead of a straight line tracing the shortest path between points, the values are connected by continuous vertical and horizontal lines forming a step like progression.

The following image shows a sample Step Area Chart.

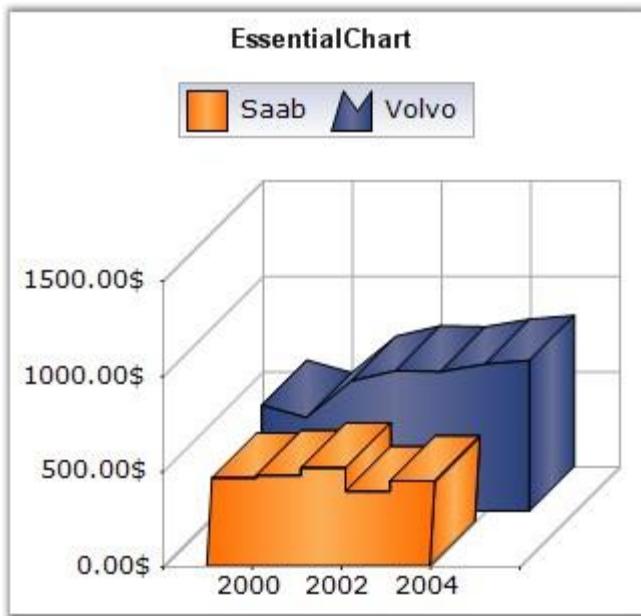


Figure 60: Chart displaying Step Area Series

### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar and Radar.

Step Area series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries ("Series
Name",ChartSeriesType.StepArea);
series.Points.Add(0, 1);
series.Points.Add(1, 3);
series.Points.Add(2, 4);
series.Points.Add(3, 2);
series.Points.Add(4, 3);

// Add the series to the chart series collection.
```

```
this.chartControl1.Series.Add (series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series  
Name", ChartSeriesType.StepArea)  
series.Points.Add(0, 1)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
series.Points.Add(3, 2)  
series.Points.Add(4, 3)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add (series)
```

**Customization Options**

Border, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Rotate, SeriesToolTipFormat, Spacing Between Series, StepItem.Inverted, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.4.6 Range Area Chart

RangeArea chart is similar to the Area charts; the only difference is, we need to give two y values (Start & End). RangeArea chart will be rendered from the start value of the x axis(Lower bounds), to end value of the y axis(upper bounds) above, on the corresponding x axis values.

This chart type gives a clear look and it may be used in cases, where we have to display range of values, per single x point. For ex: if we have to display the range of temperature per day in a chart, RangeArea Chart will be the most convenient type of chart.

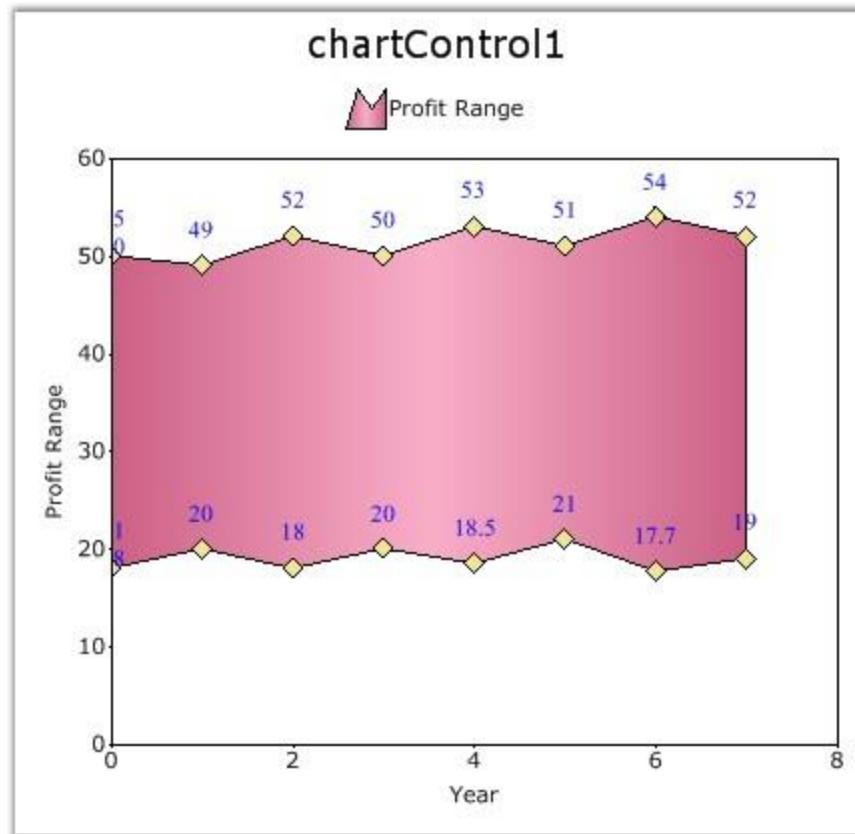


Figure 61: RangeArea Chart

### Chart Details

Details	
<b>Number of Y values per point</b>	2
<b>Maximum Number of Series</b>	Unlimited
<b>Minimum Number of Series</b>	1

Step Area series can be added to the chart using the following code.

[C#]

```
ChartSeries series1 = this.chartControl1.Model.NewSeries("Profit
Range", ChartSeriesType.RangeArea);
series1.Points.Add(0, 18, 50);
series1.Points.Add(1, 20, 49);
series1.Points.Add(2, 18, 52);
```

```
series1.Points.Add(3,20, 50);
series1.Points.Add(4, 18.5,53);
series1.Points.Add(5, 21, 51);
series1.Points.Add(6, 17.7, 54);
series1.Points.Add(7, 19, 52);
this.chartControll1.Series.Add(series1);
```

**[VB.NET]**

```
Dim series1 As ChartSeries = Me.chartControll1.Model.NewSeries ("Profit
Range", ChartSeriesType.RangeArea)
series1.Points.Add(0, 18, 50)
series1.Points.Add(1,20,49)
series1.Points.Add(2,18 , 52)
series1.Points.Add(3,20, 50)
series1.Points.Add(4, 18.5,53)
series1.Points.Add(5, 21, 51)
series1.Points.Add(6, 17.7, 54)
series1.Points.Add(7, 19, 52)

Me.chartControll1.Series.Add (series1)
```

**Customization Options**

Border, DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders,  
HighlightInterior, ImageIndex, Rotate, SeriesToolTipFormat

Spacing Between Series, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat,  
SmartLabels,

Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

## 4.4.5 Accumulation Charts

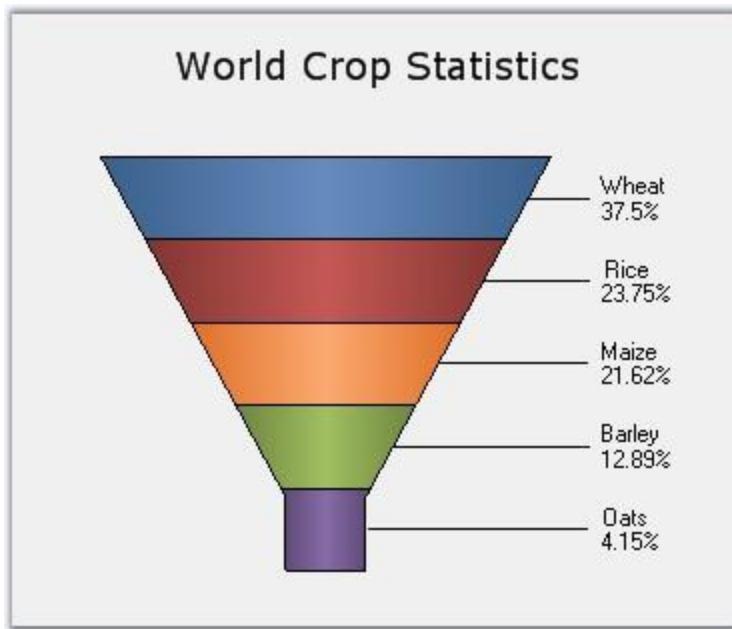
Accumulation charts are typically single series charts representing the data as portions of 100% and do not use any axes. Essential Chart offers the two types of Accumulation charts.

### 4.4.5.1 Funnel Chart

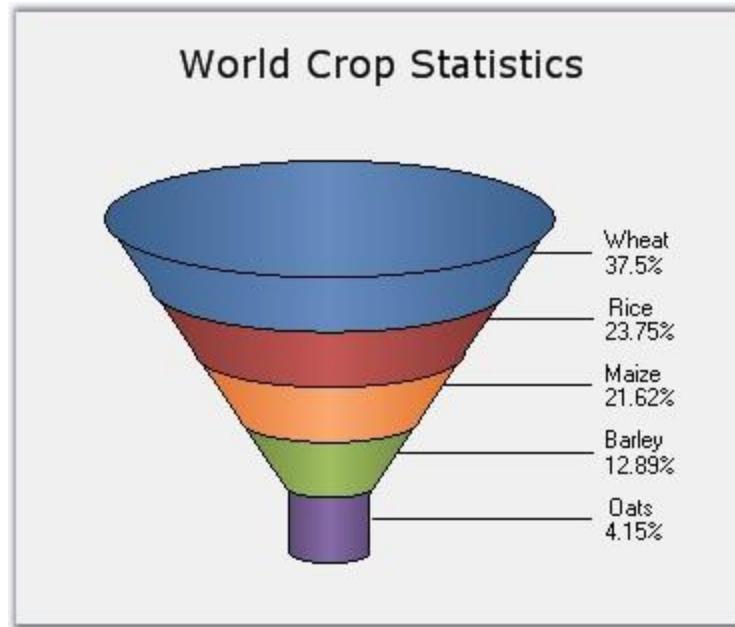
The Funnel chart is a single series chart representing the data as portions of 100%, and this chart does not use any axes. Funnel chart can be viewed in 2D or 3D mode.

Funnel charts are often used to represent stages in a sales process and show the amount of potential revenue for each stage. This type of chart can be useful also in identifying potential problem areas in an organization's sales processes. A funnel chart is similar to a stacked percent bar chart.

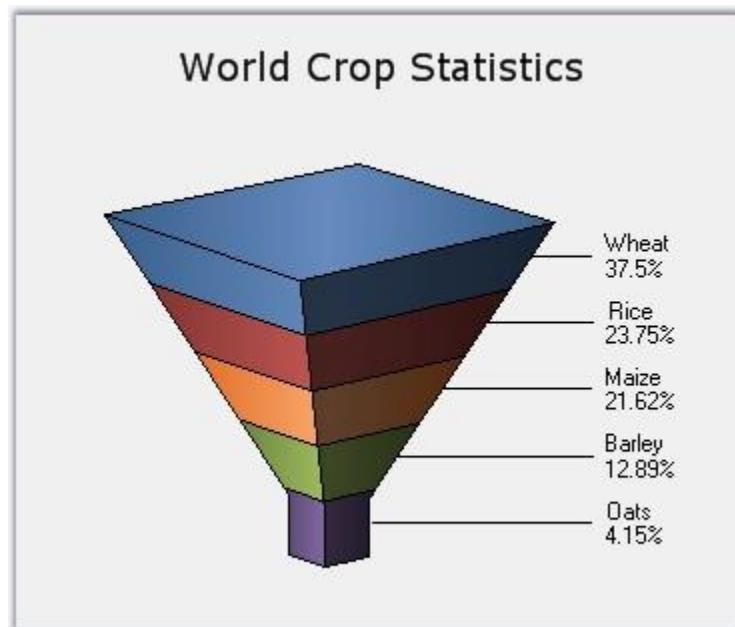
The following images are some sample Funnel Charts.



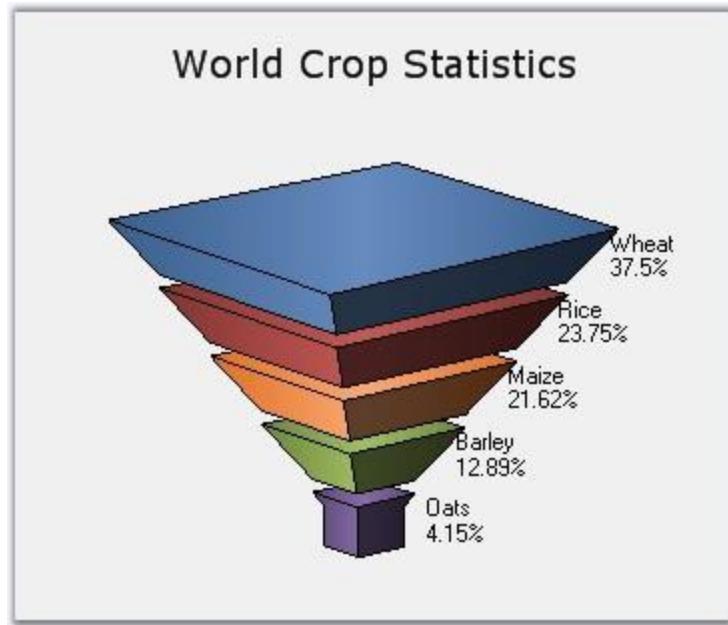
*Figure 62: 2DFunnelChart*



*Figure 63: 3D Funnel-FigureBase-Circle Chart*



*Figure 64: 3D Funnel-FigureBase-Square Chart*



*Figure 65: 3D Funnel Chart with Gap ratio 0.1*

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One.
<b>Cannot be Combined with</b>	Any other chart types.

[C#]

```
ChartSeries series1 = this.chartControl1.Model.NewSeries("Funnel chart", ChartSeriesType.Funnel);
series1.Points.Add(0, 25.3);
series1.Points.Add(1, 45.7);
series1.Points.Add(2, 97.3);
series1.Points.Add(3, 20.6);
series1.Points.Add(4, 125.8);
```

```
series1.Points.Add(5, 216.1);
this.chartControl1.Series.Add(series1);
```

**[VB.NET]**

```
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("Funnel
chart", ChartSeriesType.Funnel)
series1.Points.Add(0,25.3)
series1.Points.Add(1,45.7)
series1.Points.Add(2,97.3)
series1.Points.Add(3,20.6)
series1.Points.Add(4,125.8)
series1.Points.Add(5,216.1)
Me.chartControl1.Series.Add(series1)
```

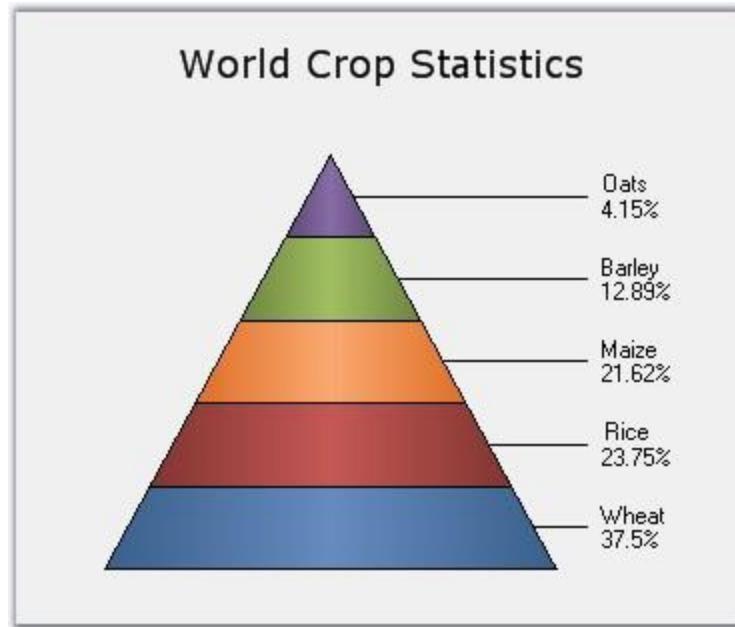
**Customization Options**

Border, DisplayText, DrawSeriesNameInDepth, FigureBase, FunnelMode, GapRatio, HighlightInterior, LabelPlacement, LabelStyle, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible, ShowDataBindLabels

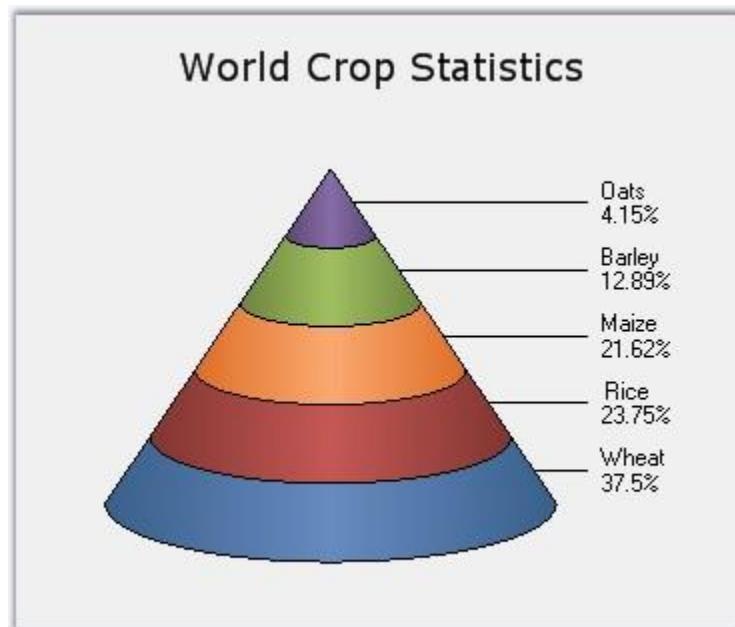
#### 4.4.5.2 Pyramid Chart

Pyramid chart is similar to the funnel chart. It's often used for geographical purposes. The Pyramid Chart type displays the data, which when totalled will be 100%. This type of chart is a single series chart representing the data as portions of 100%, and this chart does not use any axes. Pyramid chart can be viewed as 2D or 3D.

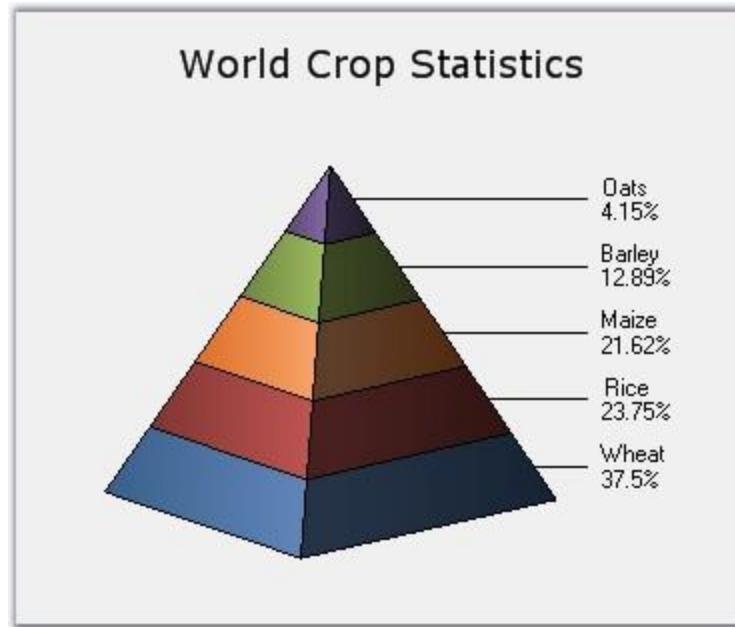
The following images are some sample Pyramid Charts.



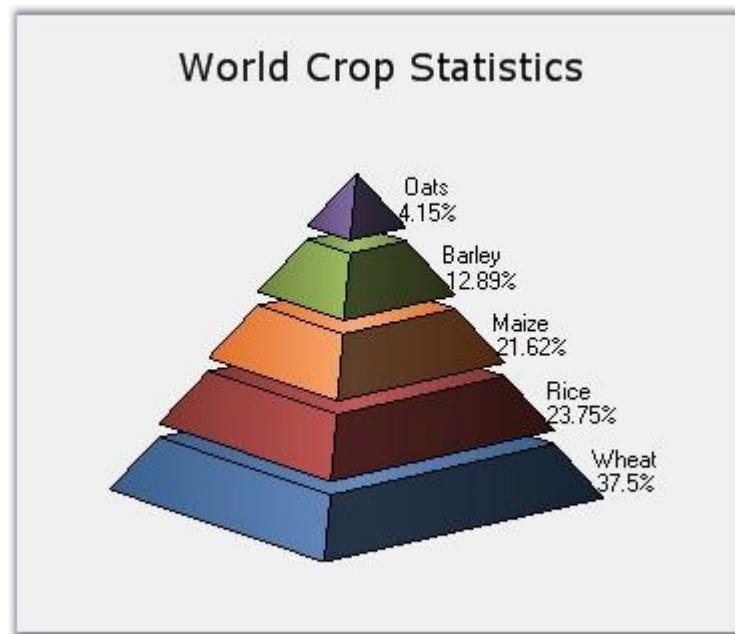
*Figure 66: 2D Pyramid Chart*



*Figure 67: 3D Pyramid-FigureBase-Circle Chart*



*Figure 68: 3D Pyramid-FigureBase-Square Chart*



*Figure 69: 3D Pyramid Chart with Gap ratio 0.1*

Details	
Number of Y values per point	1

<b>Number of Series</b>	One.
<b>Cannot be Combined with</b>	Any other chart types.

**[C#]**

```
ChartSeries series1 = this.chartControl1.Model.NewSeries("Pyramid  
chart", ChartSeriesType.Pyramid);  
series1.Points.Add(0, 25.3);  
series1.Points.Add(1, 45.7);  
series1.Points.Add(2, 97.3);  
series1.Points.Add(3, 20.6);  
series1.Points.Add(4, 125.8);  
series1.Points.Add(5, 216.1);  
this.chartControl1.Series.Add(series1);
```

**[VB .NET]**

```
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("Pyramid  
chart", ChartSeriesType.Pyramid)  
series1.Points.Add(0,25.3)  
series1.Points.Add(1,45.7)  
series1.Points.Add(2,97.3)  
series1.Points.Add(3,20.6)  
series1.Points.Add(4,125.8)  
series1.Points.Add(5,216.1)  
Me.chartControl1.Series.Add(series1)
```

**Customization Options**

Border, DisplayText, DrawSeriesNameInDepth, FigureBase, GapRatio, HighlightInterior,  
LabelPlacement, LabelStyle, PyramidMode, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset,  
TextOrientation, Visible, ShowDataBindLabels

#### 4.4.6 XY Charts (Bubble and Scatter)

Also known as XY Charts, these charts are used to visualize the relationship between two variables that relate to the same event.

More details on the corresponding chart types:

#### 4.4.6.1 Scatter Chart

Scatter Charts, also known as XY Charts, are a plot of y values and x values along the two axes. The points are not joined together and can be customized using shapes or images to make them easily identifiable, usually independent of time.

The scatter graph lets you plot data points based on two independent variables. The variable that we seek to predict is called the dependent variable or **y-variable**. The variable on which it depends is called the independent variable or the **x-variable**. Scatter graphs can chart multiple data sets, each represented by a different symbol and each having any number of data points.

It is used to display numerical data, either discrete or continuous. Scatter charts are commonly used for visualizing scientific data.

The following image shows a multi series Scatter Chart.

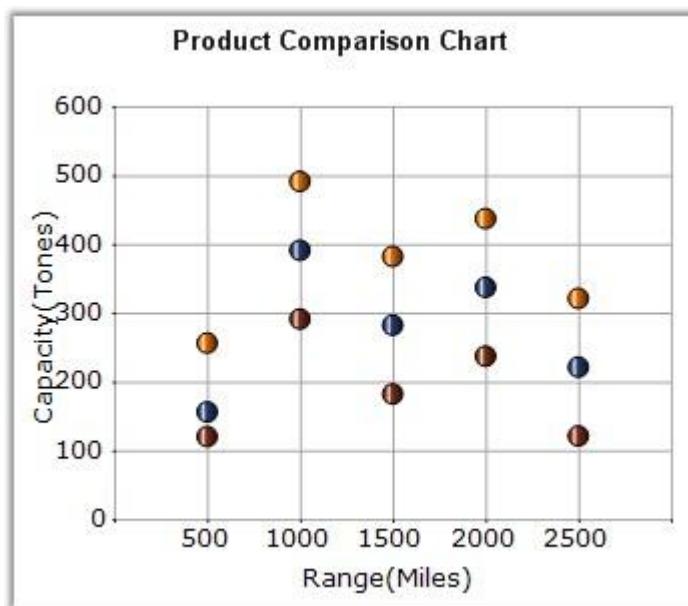


Figure 70: Chart Displaying Scatter Series

#### Chart Details

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar.

Scatter series can be added to the chart using the following code.

#### [C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries ("Series
Name",ChartSeriesType.Scatter);
series.Points.Add (0, 1);
series.Points.Add (1, 3);
series.Points.Add (2, 4);

// Add the series to the chart series collection.
this.chartControl1.Series.Add (series);
```

#### [VB .NET]

```
' Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series
Name", ChartSeriesType.Scatter)
series.Points.Add (0, 1)
series.Points.Add (1, 3)
series.Points.Add (2, 4)

' Add the series to the chart series collection.
Me.chartControl1.Series.Add (series)
```

The symbols can be configured using the **ChartSeries.Styles[i].Symbol** property as in the following example.

#### [C#]

```
// Specify the symbol info required for the Scatter chart.
series.Styles [0].Symbol = new ChartSymbolInfo();
series.Styles [0].Symbol.Color = Color.Red;
series.Styles [0].Symbol.Shape = ChartSymbolShape.InvertedTriangle;

series.Styles [1].Symbol = new ChartSymbolInfo();
```

```
series.Styles [1].Symbol.Color = Color.Green;
series.Styles [1].Symbol.Shape = ChartSymbolShape.Hexagon;

series.Styles [2].Symbol = new ChartSymbolInfo();
series.Styles [2].Symbol.Color = Color.Blue;
series.Styles [2].Symbol.Shape = ChartSymbolShape.Cross;
```

**[VB.NET]**

```
' Specify the symbol info required for the Scatter chart.
series.Styles (0).Symbol = New ChartSymbolInfo()
series.Styles (0).Symbol.Color = Color.Red
series.Styles (0).Symbol.Shape = ChartSymbolShape.InvertedTriangle

series.Styles (1).Symbol = New ChartSymbolInfo()
series.Styles (1).Symbol.Color = Color.Green
series.Styles (1).Symbol.Shape = ChartSymbolShape.Hexagon

series.Styles (2).Symbol = New ChartSymbolInfo()
series.Styles (2).Symbol.Color = Color.Blue
series.Styles (2).Symbol.Shape = ChartSymbolShape.Cross
```

**Customization Options**

DisplayText, DrawSeriesNameInDepth, LightAngle, LightColor, PhongAlpha, ScatterConnectType, ScatterSplineTension, ToolTip, ToolTipFormat, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

**See Also**

[Bubble Chart](#)

#### 4.4.6.2 Bubble Chart

Bubble Chart is an extension of the Scatter Chart (or XY-chart) where each data marker is represented by a circle whose dimension form a third variable. Consequently, bubble charts allow three-variable comparisons allowing for easy visualization of complex interdependencies that are not apparent in two-variable charts. Bubble charts are frequently used in market and product comparison studies.

Though it's called a bubble chart, the data marker can be rendered as either a circle, image or square using the **BubbleType** property.

The following image shows a multi series Bubble Chart.

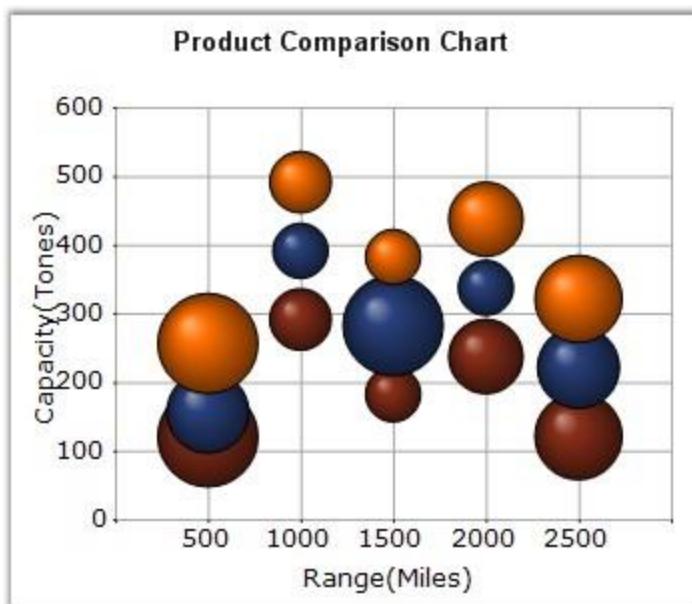


Figure 71: Chart Displaying 3 Bubble Series

### Chart Details

Details	
<b>Number of Y values per point</b>	2 (optional second value defines the size of the shape).
<b>Number of Series</b>	One or More
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar.

Bubble series can be added to the chart using the following code.

```
[C#]

// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.Bubble);
// The 2nd Y value represents the size of the shape
```

```
series.Points.Add (0, 1, 7);
series.Points.Add (1, 3, 5);
series.Points.Add (2, 4, 9);

// Add the series to the chart series collection.
this.chartControll1.Series.Add (series);
```

**[VB.NET]**

```
' Create chart series and add data point into it.
Dim series As ChartSeries = Me.chartControll1.Model.NewSeries("Series
Name", ChartSeriesType.Bubble)
' The 2nd Y value represents the size of the shape.
series.Points.Add (0, 1, 7)
series.Points.Add (1, 3, 5)
series.Points.Add (2, 4, 9)

' Add the series to the chart series collection.
Me.chartControll1.Series.Add (series)
```

### Customization Options

Border, BubbleType, DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, EnablePhongStyle, HighlightInterior, ImageIndex

Images, Spacing Between Series, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

### See Also

[Scatter Chart](#)

## 4.4.7 Financial Charts

The following charts are a staple of analytical reports in the financial world. Financial data usually has more than one y value. For example, stock price charts should include high, low, open and close prices for a day. Such data needs to be appropriately rendered in the context of "stock market data". Also, besides actual values, trends in price movement need to be depicted visually.

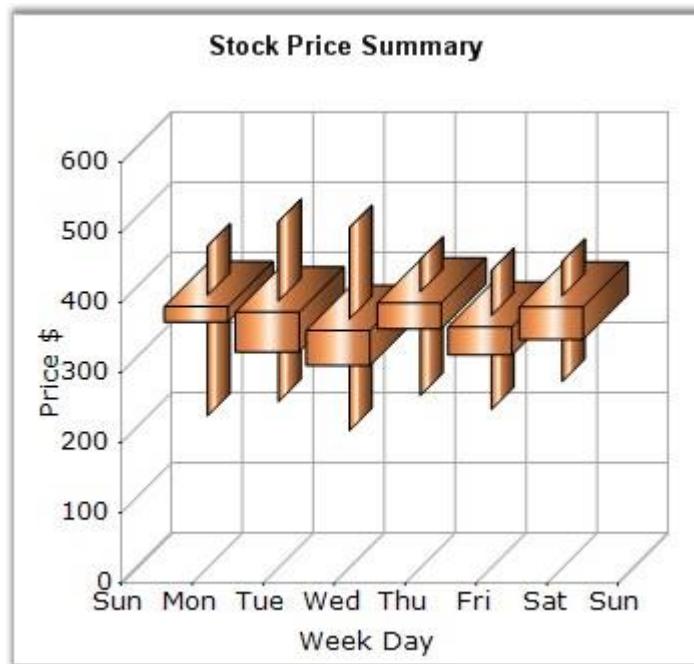
The following charts cater to the above requirements and provide an easy-to-decipher picture of price action.

Financial Chart types supported in Essential Chart.

#### 4.4.7.1 Candle Chart

A Candle chart displays stock information using the High, Low, Open and Close values. The Hi and Lo values are represented by the wick of a candle. The candle represents open and close values.

The following image shows a CandleChart displaying a single series.



*Figure 72: Chart displaying Candle Series*

#### Chart Details

Details	
Number of Y values per point	4 (High, Low , Open and Close respectively).

<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar.

Candle series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data point into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name",ChartSeriesType.Candle);  
// Arguments: X value, High, Low, Open, Close  
series.Points.Add(0, 5, 1, 3, 4);  
series.Points.Add(1, 8, 7, 4, 7);  
series.Points.Add(2, 8, 4, 5, 6);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB .NET]**

```
' Create chart series and add data point into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Candle)  
' Arguments: X value, High, Low, Open, Close  
series.Points.Add(0, 5, 1, 3, 4)  
series.Points.Add(1, 8, 7, 4, 7)  
series.Points.Add(2, 8, 4, 5, 6)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

### Customization Options

Border, DisplayShadow, DisplayText, DrawSeriesNameInDepth, ImageIndex, Images, PhongAlpha, Rotate, Spacing Between Series

ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.7.2 Hi Lo Chart

Hi Lo Chart is a special kind of chart that is normally used in stock analysis. They are typically used to display error bars or the trading range of a stock for each period.

The Hi Lo Chart expect two y values to be specified in the series. One value should represent the high and the other value should represent the low stock price for the period. This can be specified in any order.

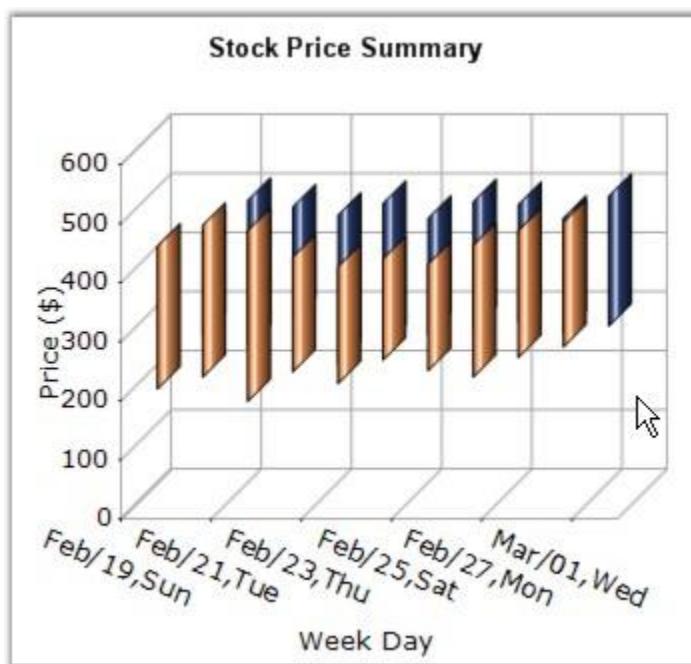


Figure 73: Chart displaying Hi Lo Series

#### Chart Details

Details	
<b>Number of Y values per point</b>	2
<b>Number of Series</b>	One or More

<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar
--------------------------------	--------------------------------------------

Hi Lo series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.HiLo);  
series.Points.Add(0, 1, 3);  
series.Points.Add(1, 3, 4);  
series.Points.Add(2, 4, 8);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.HiLo)  
series.Points.Add(0, 1, 3)  
series.Points.Add(1, 3, 4)  
series.Points.Add(2, 4, 8)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

**Customization Options**

DisplayText, DrawErrorBars, DrawSeriesNameInDepth, ErrorBarsSymbolShape, PhongAlpha, Rotate, Spacing Between Series, ShadingMode, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.7.3 Hi Lo Open Close Chart

Hi Lo Open Close Chart is a special kind of chart that is normally used in stock analysis. This chart type expects four y values for every point in the series. Those values should represent the High, Low, Open and Close values of the stock, in that order, for that period.

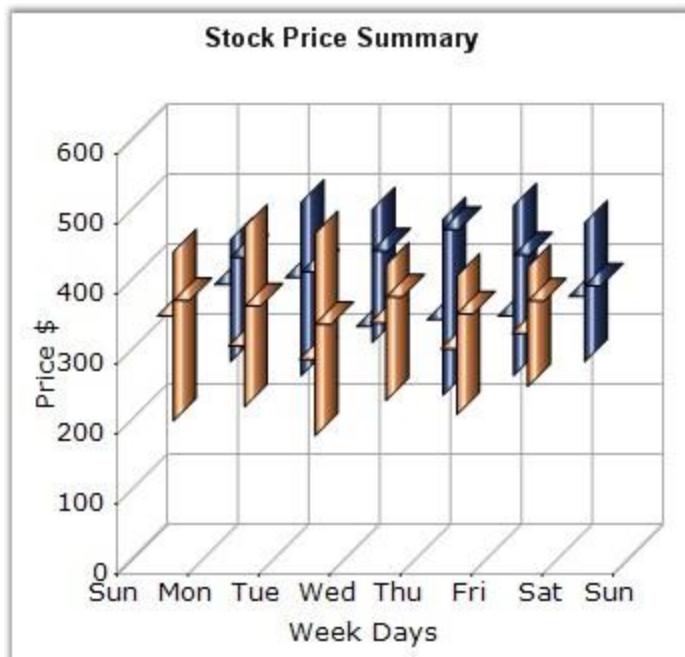


Figure 74: Chart displaying Hi Lo Open Close Series

### Chart Details

Details	
<b>Number of Y values per point</b>	4.
<b>Number of Series</b>	One or More.
<b>Cannot be Combined with</b>	Pie, Bar, Stacked Bar charts, Polar, Radar

Hi Lo Open Close series can be added to the chart using the following code.

```
[C#]

// Create chart series and add data point into it.
ChartSeries series = this.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.HiLoOpenClose);
// Arguments: X value, High, Low, Open, Close
series.Points.Add(0, 5, 1, 3, 4);
series.Points.Add(1, 8, 7, 4, 7);
series.Points.Add(2, 8, 4, 5, 6);
```

```
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB .NET]**

```
' Create chart series and add data point into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.HiLoOpenClose)  
' Arguments: X value, High, Low, Open, Close  
series.Points.Add(0, 5, 1, 3, 4)  
series.Points.Add(1, 8, 7, 4, 7)  
series.Points.Add(2, 8, 4, 5, 6)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

**Customization Options**

DisplayText, DrawSeriesNameInDepth, OpenCloseDrawMode, PhongAlpha, Rotate, Spacing  
Between Series, ShadingMode, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset,  
TextOrientation, Visible

#### 4.4.7.4 Kagi Chart

Kagi Charts are a Japanese invention and date since the late 1870's, but were popularized in the western world by Steven Nison. They contain a series of connecting vertical lines where the thickness and direction of those lines depend on price. If closing prices continue to move in the direction of the prior vertical Kagi line, then that line is extended. However, if the closing price reverses by a pre-determined "reversal" amount, a new Kagi line is drawn in the next column in the opposite direction.

The penetration of a prior column's high or low, by the latest closing price, alters the colors of the lines. These colors depict either a bullish or bearish pattern. Use the **PriceUpColor** and **PriceDownColor** properties to specify the colors for these two patterns. The wider the columns, the stronger the pattern.

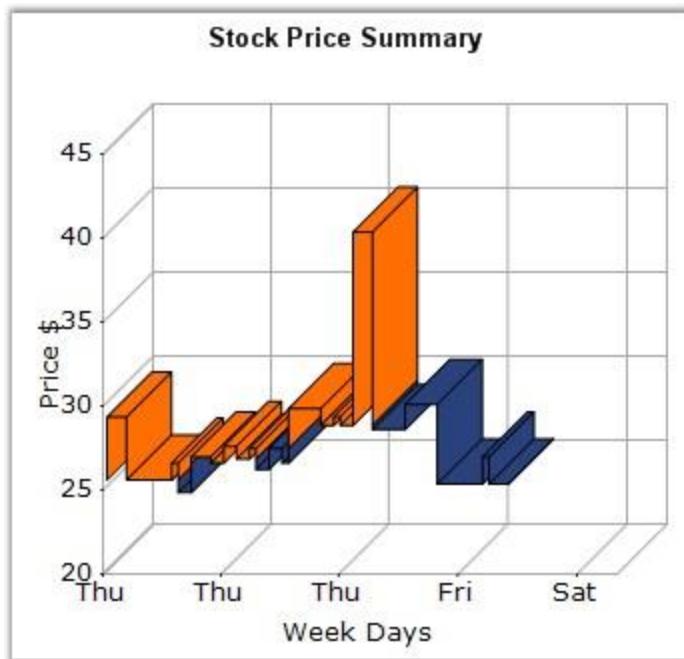


Figure 75: Chart displaying Kagi Series

#### Chart Details

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One.
<b>Cannot be Combined with</b>	Pie, Bar.

Kagi series can be added to the chart using the following code.

```
[C#]

// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries ("Series Name",
ChartSeriesType.Kagi);
// Arguments: X value, closing price.
```

```
series.Points.Add(0, 23);
series.Points.Add(1, 27);
series.Points.Add(2, 24.7);
series.Points.Add(3, 23);
series.Points.Add(4, 21);
series.Points.Add(5, 20);
series.Points.Add(6, 22);
series.Points.Add(7, 24);
series.Points.Add(8, 26);

series.Text = series.Name;
series.ReversalAmount = 1.0;
series.PriceUpColor = Color.Green;
series.PriceDownColor = Color.Red;

// Add the series to the chart series collection.
this.chartControl1.Series.Add (series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series
Name", ChartSeriesType.Kagi)
' Arguments: X value, closing price.
series.Points.Add(0, 23)
series.Points.Add(1, 27)
series.Points.Add(2, 24.7)
series.Points.Add(3, 23)
series.Points.Add(4, 21)
series.Points.Add(5, 20)
series.Points.Add(6, 22)
series.Points.Add(7, 24)
series.Points.Add(8, 26)

series.Text = series.Name
series.ReversalAmount = 1.0
series.PriceUpColor = Color.Green
series.PriceDownColor = Color.Red

' Add the series to the chart series collection.
Me.chartControl1.Series.Add (series)
```

If the ReversalAmount is 0.0 instead of 1.0 which is the default value, then the Kagi chart will look like the below image.

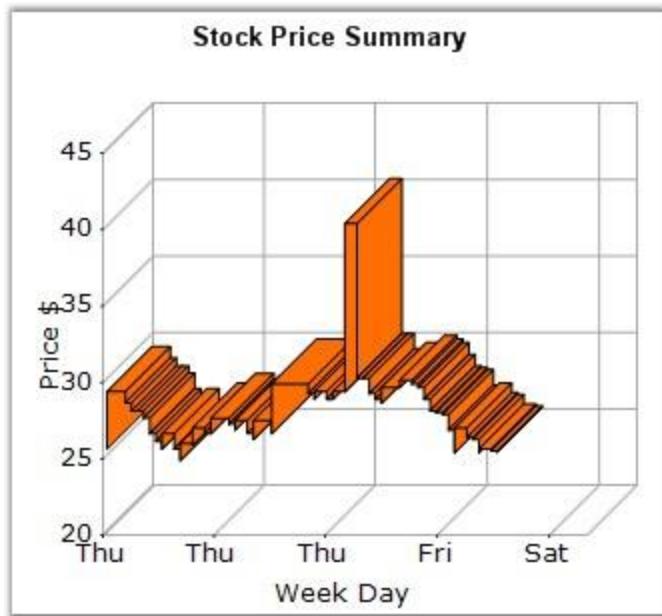


Figure 76: Kagi Chart with ReversalAmount set to 0.0

#### Customization Options

DisplayShadow, DisplayText, DrawSeriesNameInDepth, PriceDownColor, PriceUpColor, ReversalAmount, Rotate, Spacing Between Series

ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.7.5 Point and Figure Chart

Point and Figure Chart is used to identify support levels, resistance levels and chart patterns. The chart ignores the time factor and concentrates solely on movements in price - a column of Xs or Os may take one day or several weeks to complete. By convention, the first X in a column is plotted one box above the last O in the previous column (and the first O in a column is plotted one box below the highest X).

This is a chart that plots the day-to-day increment and decrement in price. It uses a series of Xs and Os to determine price trends where the Xs represent an upward trend and the Os represent a downward trend. The default value of **ReversalAmount** is 1. Use the **PriceUpColor** to specify the color for the Xs and **PriceDownColor** to specify the color for the Os.

This chart requires two y values, the high value and the low value for the specified period.

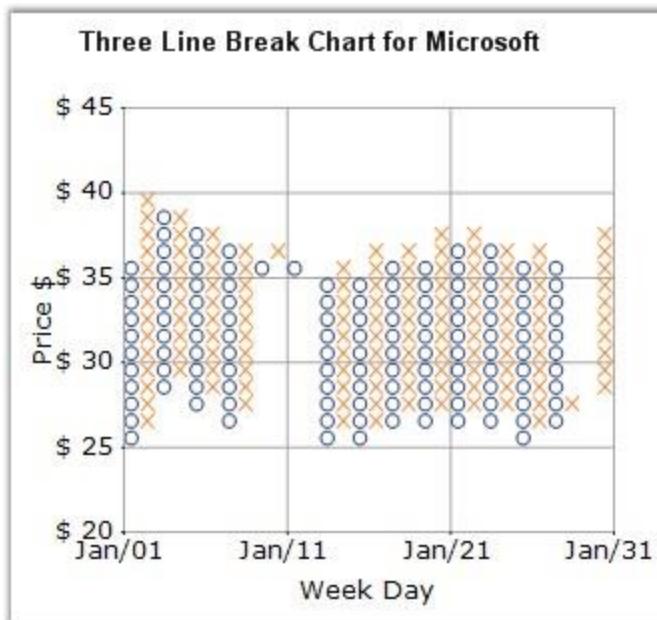


Figure 77: Chart displaying Point And Figure Series

### Chart Details

Details	
<b>Number of Y values per point</b>	2
<b>Number of Series</b>	One
<b>Cannot be Combined with</b>	Pie, Bar

Point and Figure series can be added to the chart using the following code.

[C#]

```
// Create chart series and add data points into it.
ChartSeries series = this.chartControl1.Model.NewSeries ("Series
Name",ChartSeriesType.PointAndFigure);
// Arguments: X value, low value, high value
series.Points.Add (0, 1, 5);
series.Points.Add (1, 3, 7);
```

```
series.Points.Add (2, 4, 8);
series.ReversalAmount = 1.0;

// Add the series to the chart series collection.
this.chartControl1.Series.Add (series);
```

**[VB .NET]**

```
' Create chart series and add data points into it.
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series
Name",ChartSeriesType.PointAndFigure)
' Arguments: X value, low value, high value
series.Points.Add (0, 1, 5)
series.Points.Add (1, 3, 7)
series.Points.Add (2, 4, 8)
series.ReversalAmount = 1.0

' Add the series to the chart series collection.
Me.chartControl1.Series.Add (series)
```

#### Customization Options

DisplayShadow, DisplayText, DrawSeriesNameInDepth, HeightBox, PriceDownColor, PriceUpColor, ReversalAmount, Spacing Between Series

ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

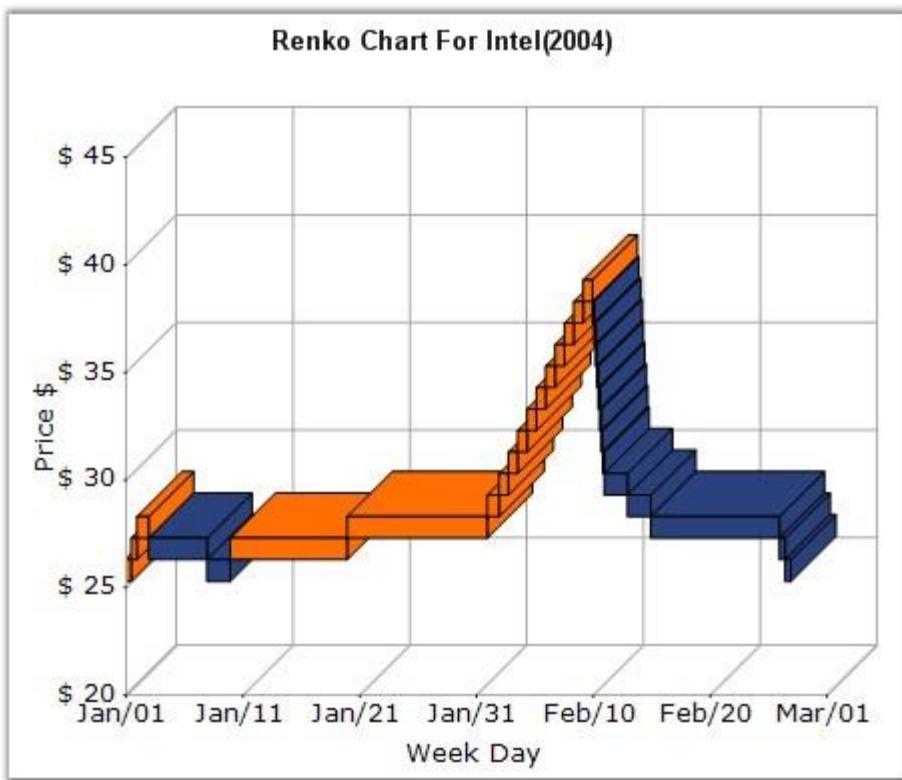
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.7.6 Renko Chart

Renko charting method is thought to have acquired its name from "renga" which is the Japanese word for bricks. Renko Charts were introduced by Steve Nison. Renko (Bricks) are drawn equal in size for a determined amount. A brick is drawn in the direction of the prior move only if prices move by a minimum amount. If prices change by the determined amount or more, a new brick is drawn. If prices change by less than the determined amount (specified by **ReversalAmount**), the new price is ignored. The default value of ReversalAmount is 1.

If the new closing price penetrates the previous bricks closing price in the opposite direction a trend reversal highlighted by the change in color of the bricks happens. Use the **PriceUpColor** to indicate bullish trend and **PriceDownColor** to indicate bearish trend.

Since a Renko chart isolates the underlying trends by filtering out the minor ups and downs, Renko charts are excellent in determining support and resistance levels.



*Figure 78: Chart displaying Renko Series*

#### **Chart Details**

Details	
<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One

<b>Cannot be Combined with</b>	Pie, Bar
--------------------------------	----------

Renko series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries ("Series  
Name",ChartSeriesType.Renko);  
series.Points.Add (0, 1);  
series.Points.Add (1, 3);  
series.Points.Add (2, 4);  
series.ReversalAmount = 1.0;  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add (series);
```

**[VB .NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series  
Name",ChartSeriesType.Renko)  
series.Points.Add (0, 1)  
series.Points.Add (1, 3)  
series.Points.Add (2, 4)  
series.ReversalAmount = 1.0  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add (series)
```

### Customization Options

Border, ColorsMode, DarkLightPower, DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images, PriceDownColor

PriceUpColor, ReversalAmount, Spacing Between Series, ShadowInterior, ShadowOffset, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormatSmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.7.7 Three Line Break Chart

Three Line Break Chart is similar in concept to point and figure charts. The Three Line Break charting method is so-named because of the number of lines typically used. It displays a series of vertical boxes ("lines") that are based on changes in prices. It ignores the passage of time.

The three-line break chart looks like a series of rising and falling lines of varying heights. Each new line, like the X's and O's of a point and figure chart, occupies a new column. Based on closing prices (or highs and lows), a new rising line is drawn if the previous high is exceeded and a new falling line is drawn if the price hits a new low. Change in price trends are highlighted by changing colors. Use the **PriceUpColor** to indicate bullish trend and **PriceDownColor** to indicate bearish trend.

The **ReversalAmount** specifies the threshold amount by which the price should change to begin rendering a new vertical box in the appropriate direction.

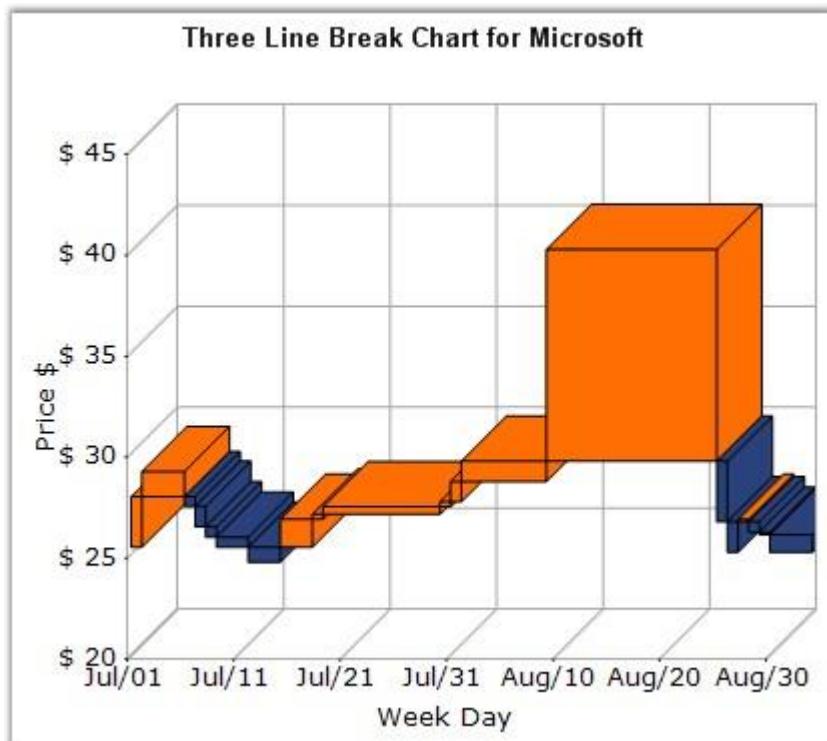


Figure 79: Chart displaying Three Line Break Series

#### Chart Details

Details
---------

<b>Number of Y values per point</b>	1
<b>Number of Series</b>	One
<b>Cannot be Combined with</b>	Pie, Bar

Three Line Break series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries ("Series  
Name", ChartSeriesType.ThreeLineBreak);  
series.Points.Add (0, 1);  
series.Points.Add (1, 3);  
series.Points.Add (2, 4);  
  
// Add series to the chart series collection.  
this.chartControl1.Series.Add (series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series  
Name", ChartSeriesType.ThreeLineBreak)  
series.Points.Add (0, 1)  
series.Points.Add (1, 3)  
series.Points.Add (2, 4)  
  
' Add series to the chart series collection.  
Me.chartControl1.Series.Add (series)
```

**Customization Options**

DisplayShadow, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images, PriceDownColor, PriceUpColor, Spacing Between Series

ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

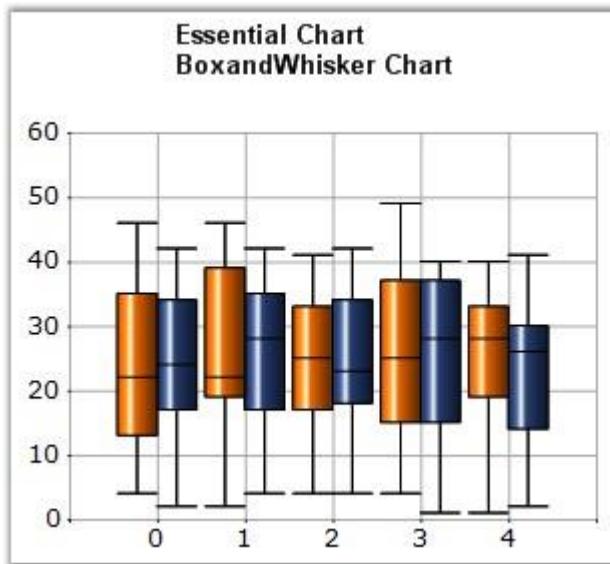
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.7.8 Box And Whisker Chart

In 1977, John Tukey published an efficient method for displaying a five-number data summary. The graph is called a Box and Whisker plot (also known as BoxPlot) and summarizes the following statistical measures.

- median
- upper and lower quartiles (75 percentile to 25 percentile)
- minimum and maximum data values

The following is an example of a Box and Whisker plot.



*Figure 80: BoxAndWhisker Chart*

The Box and Whisker plot is interpreted as follows.

- The box itself contains the middle 50% of the data. The upper edge (hinge) of the box indicates the 75th percentile of the data set and the lower hinge indicates the 25th percentile. The range of the middle two quartiles is known as the inter-quartile range.
- The line in the box indicates the median value of the data.
- Box and Whisker chart has two modes, Normal mode and Percentile mode.

- In Normal Mode, if the median line within the box is not equi-distant from the hinges, then the data is skewed. The ends of the vertical lines or "whiskers" indicate the minimum and maximum data values, unless outliers are present, in which case the whiskers extend to a maximum of 1.5 times the inter-quartile range.
- In Percentile Mode: [Set **Series1.ConfigItems.BoxAndWhiskerItem.PercentileMode** property to **true**], the ends of the vertical lines or "whiskers" will be decided by the Series1.ConfigItems.BoxAndWhiskerItem.Percentile property value. For example, if the 'Percentile' value is 0.15, then the minimum value will be the 15th percentile of the overall data set and the maximum value will be 85th percentile of the overall data set.

**Note:**

1. **The percentile value should lie between 0.0 to 0.25.**
2. **It is not possible to set upper Percentile value. It is calculated automatically based on the Percentile value.**

**For Ex:**

**Percentile = 0.15**

**Upper Percentile = 1 - Percentile = 0.85.**

**In Normal mode, Outliers are present in which case the whiskers extend to a maximum of 1.5 times the inter-quartile range. But in Percentile mode, Outliers will be calculated based on the Percentile value.**

**For example:**

**Percentile = 0.15**

**Outliers are present in which case the whiskers extend to minimum and maximum of 15th and 85th percentile of overall data set, respectively. If 'Percentile' value is Zero, then, there is no outliers in the Chart.**

3. **The width of the Outliers can be adjusted by using this 'Series1.ConfigItems.BoxAndWhiskerItem.OutlierWidth' property. If it is zero, the width of the outlier will be calculated based on the data points range.**

### **Chart Details**

<b>Details</b>	
<b>Number of Y values per point</b>	5 (minimum, lower quartile, median, upper quartile, maximum)
<b>Number of Series</b>	One or more
<b>Cannot be Combined with</b>	Pie, Bar, Polar, Radar

Box and Whisker series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series1 = this.chartControl1.Model.NewSeries("Series  
1",ChartSeriesType.BoxAndWhisker );  
series1.Points.Add(1, 5, 8 ,12, 15, 18);  
series1.Points.Add(2, 4, 6, 10, 12, 14);  
series1.Points.Add(3, 2, 4, 7, 14, 18);  
  
ChartSeries series2 = this.chartControl1.Model.NewSeries("Series  
2",ChartSeriesType.BoxAndWhisker );  
series2.Points.Add(1, 6, 9, 15, 18, 20);  
series2.Points.Add(2, 7, 9, 13, 15, 16);  
series2.Points.Add(3, 6, 8, 10, 15, 19);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series1);  
this.chartControl1.Series.Add(series2);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
1",ChartSeriesType.BoxAndWhisker)  
series1.Points.Add(1, 5, 8 ,12, 15, 18)  
series1.Points.Add(2, 4, 6, 10, 12, 14)  
series1.Points.Add(3, 2, 4, 7, 14, 18)  
  
Dim series2 As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
2",ChartSeriesType.BoxAndWhisker)  
series2.Points.Add(1, 6, 9, 15, 18, 20)  
series2.Points.Add(2, 7, 9, 13, 15, 16)  
series2.Points.Add(3, 6, 8, 10, 15, 19)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series1)  
Me.chartControl1.Series.Add(series2)
```

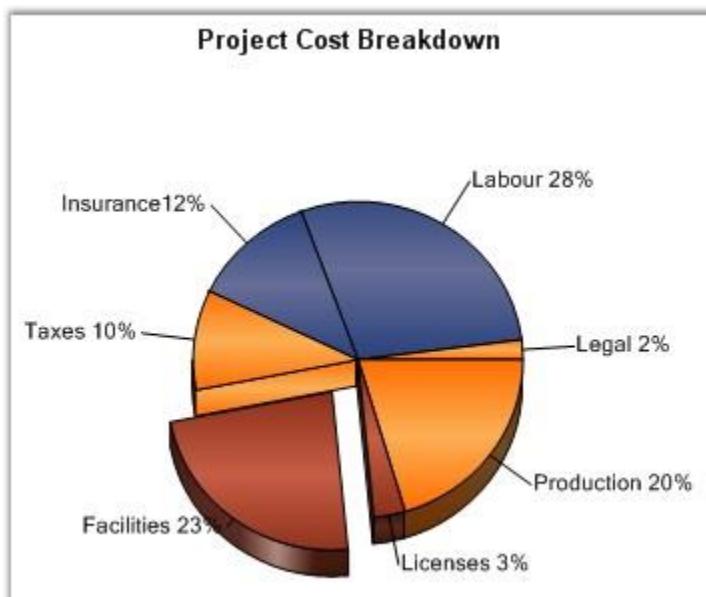
**Customization Options**

[Border](#), [ColumnDrawMode](#), [DisplayShadow](#), [DisplayText](#), [ElementBorders](#), [HighlightInterior](#),

ImageIndex, Images, LightAngle, LightColor, PhongAlpha, Rotate, Spacing, Spacing Between Series, ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.8 Pie Chart

A Pie Chart renders y values as slices in a pie. These slices are rendered in proportion to the whole, which is simply the sum of all the y values in the series. Consequently, Pie Charts are used to visualize the proportional contribution (in terms of percentage or fraction) of categories of data to the whole data set. The x values in the data series will only be treated as nominal (categorical, qualitative) data. The Pie Chart can display only one Data Series at a time.



*Figure 81: Chart displaying Pie Series*

##### Chart Details

Details	
Number of Y values per point	1
Number of Series	One.

<b>Cannot be Combined with</b>	Any other chart types.
--------------------------------	------------------------

Pie series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Pie);  
series.Points.Add(0, 1);  
series.Points.Add(1, 3);  
series.Points.Add(2, 4);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series);
```

**[VB .NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series  
Name", ChartSeriesType.Pie)  
series.Points.Add(0, 1)  
series.Points.Add(1, 3)  
series.Points.Add(2, 4)  
  
// Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series)
```

### Customization Options

AngleOffset, Border, DisplayShadow, DisplayText, DoughnutCoeficient,  
DrawSeriesNameInDepth, ElementBorders, ExplodedAll, ExplodedIndex, ExplosionOffset  
FillMode, Gradient, HeightByAreaDepth, HeightCoeficient, HighlightInterior, InSideRadius,  
OptimizePiePointPositions, PieType, ShadowInterior, ShadowOffset  
ShowTicks, VisibleAllPies, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels,  
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible,  
ShowDataBindLabels

### See Also

#### 4.4.8.1 Doughnut Chart

##### DoughnutCoefficient

PieCharts specified with a **DoughnutCoefficient** will be rendered as the Doughnut chart. By default, this value is set to 0.0 and hence the chart will be rendered as a full pie. The DoughnutCoefficient property specifies the fraction of radius occupied by the doughnut whole. Hence the value can range from 0.0 to 0.9.

##### [C#]

```
this.chartControl1.Series(0).ConfigItems.PieItem.DoughnutCoefficient=0.5f;
```

##### [VB .NET]

```
Me.chartControl1.Series(0).ConfigItems.PieItem.DoughnutCoefficient=0.5f
```

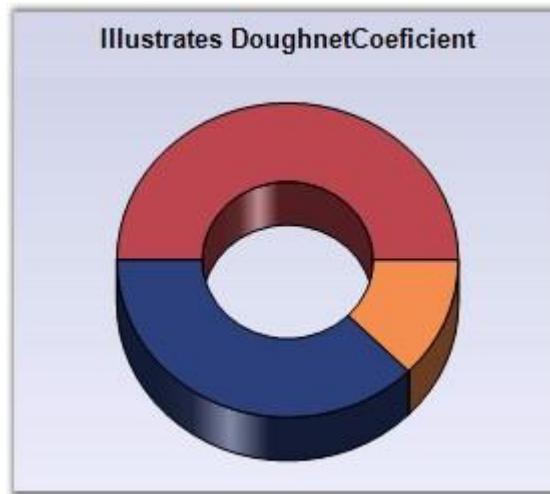


Figure 82: Pie Chart with DoughnetCoefficient Property Set

##### HeightCoefficient

When in **3D** mode, the relative height of the pie chart can be specified via the **HeightCoefficient** property. Note that the **HeightByAreaDepth** property should be set as **false** for this to take effect. The valid values are 0.1f to 0.5f. This property is set to **0.2f by default**.

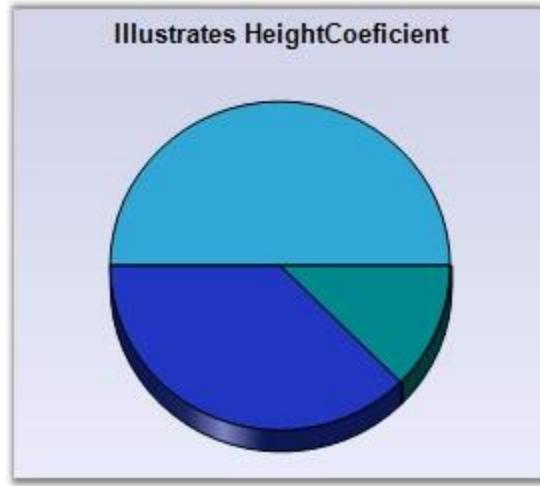
##### [C#]

```
this.chartControl1.Series[0].ConfigItems.PieItem.HeightByAreaDepth =
```

```
false;  
this.chartControl1.Series[0].ConfigItems.PieItem.HeightCoeficient =  
0.1f;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.PieItem.HeightByAreaDepth =  
False  
Me.chartControl1.Series(0).ConfigItems.PieItem.HeightCoeficient=0.1f
```



*Figure 83: Pie Chart with HeightCoeficient Property Set*

**Customization Options**

AngleOffset, Border, DisplayShadow, DisplayText, DoughnutCoeficient,  
DrawSeriesNameInDepth, ElementBorders, ExplodedAll, ExplodedIndex, ExplosionOffset  
FillMode, Gradient, HeightByAreaDepth, HeightCoeficient, HighlightInterior, InSideRadius,  
OptimizePiePointPositions, PieType, ShadowInterior, ShadowOffset  
ShowTicks, VisibleAllPies, FancyToolTip, Font, Interior, LegendItem, Name,  
PointsToolTipFormat, SmartLabels,  
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible,  
ShowDataBindLabels

#### 4.4.9 Polar And Radar Chart

Essential chart supports the implementation of Polar and Radar charts in the chart control. These charts can be used to display different values and angles in the form of a graph.

#### 4.4.9.1 Polar Chart

A Polar Chart is a circular graph on which data is displayed, in terms of values and angles. The x values define the angles at which the data points will be plotted. The y value defines the distance of the data points from the center of the graph, with the center of the graph usually starting at 0.

It is a form of graph that allows a visual comparison between several quantitative or qualitative aspects of a situation and also allows a visual comparison between several situations that are drawn using the same axes (poles).

Polar charts supports plotting the axis values in the reverse direction / clockwise direction also, by setting the **Inversed** property of axis to **true**.

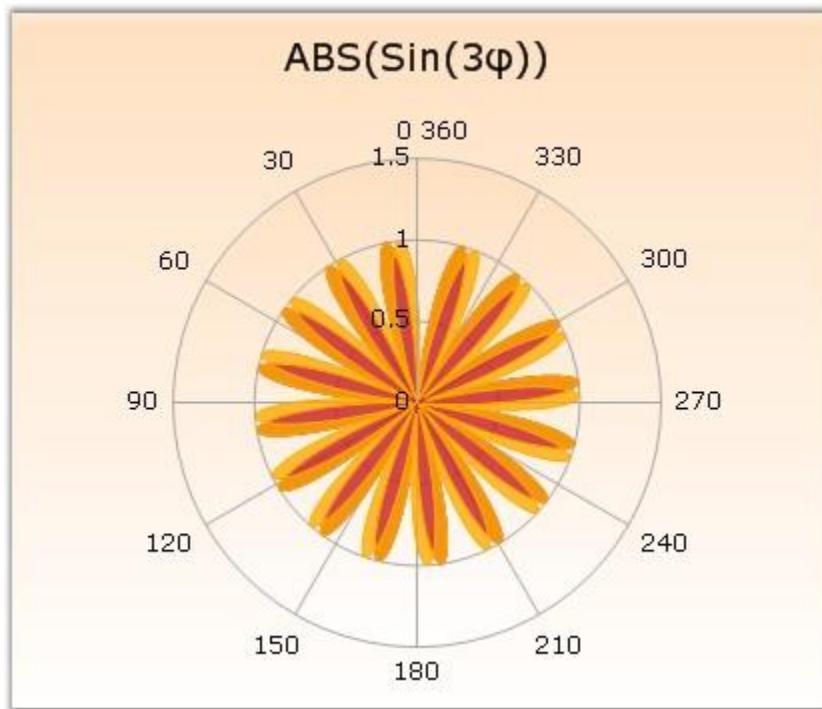


Figure 84: Chart displaying Polar Series

#### Chart Details

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One.
<b>Cannot be Combined with</b>	Any other chart types.

Polar series can be added to the chart using the following code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries ("Series  
Name",ChartSeriesType.Polar);  
series.Points.Add (0, 1);  
series.Points.Add (1, 3);  
series.Points.Add (2, 4);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add (series);
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series  
Name",ChartSeriesType.Polar)  
series.Points.Add (0, 1)  
series.Points.Add (1, 3)  
series.Points.Add (2, 4)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add (series)
```

**Customization Options**

Border, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images, LightAngle, LightColor, Radar Type, Rotate

ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,

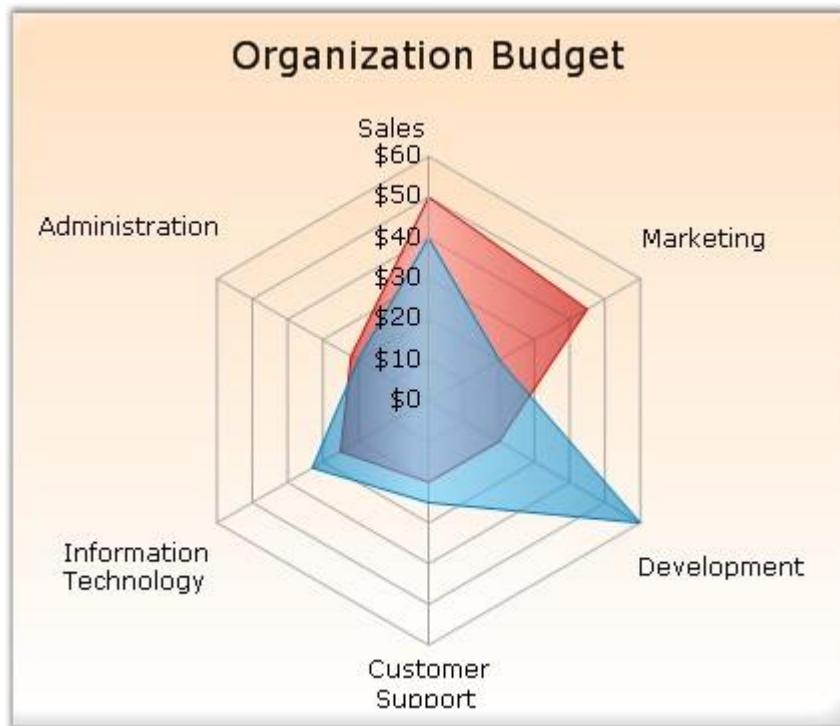
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.9.2 Radar Chart

Radar Chart is the clock face form of a line chart. It represents each data series as a line around a central point. The category (x) variable is plotted at equally spaced points around the clock. The y variable is plotted as a radius, so each category has its own y-axis radiating from the center.

##### **Some scenarios where this chart type could be used**

- When you want to compare the aggregate values of a number of data series.
- Graphically display the differences between actual and ideal performance, thereby using this chart to define performance and identifying strengths and weaknesses.
- This is also an ideal chart to use when the categories have a natural cyclic order, for example, seasons of the year.
- Radar charts supports plotting the axis values in the reverse direction / clockwise direction also, by setting the **Inversed** property of axis to **true**.



*Figure 85: Chart displaying Radar Series*

### Chart Details

Details	
<b>Number of Y values per point</b>	1.
<b>Number of Series</b>	One.
<b>Cannot be Combined with</b>	Any other chart types.

Radar series can be added to the chart using the following code.

#### [C#]

```
// Create chart series and add data points into it.  
ChartSeries series = this.chartControl1.Model.NewSeries ("Series  
Name",ChartSeriesType.Radar);  
series.Points.Add(1, 83);  
series.Points.Add(2, 79);  
series.Points.Add(3, 48);  
series.Points.Add(4, 46);  
series.Points.Add(5, 42);  
  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add (series);
```

#### [VB .NET]

```
' Create chart series and add data points into it.  
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series  
Name",ChartSeriesType.Radar)  
series.Points.Add(1, 83)  
series.Points.Add(2, 79)  
series.Points.Add(3, 48)  
series.Points.Add(4, 46)  
series.Points.Add(5, 42)  
  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add (series)
```

### Customization Options

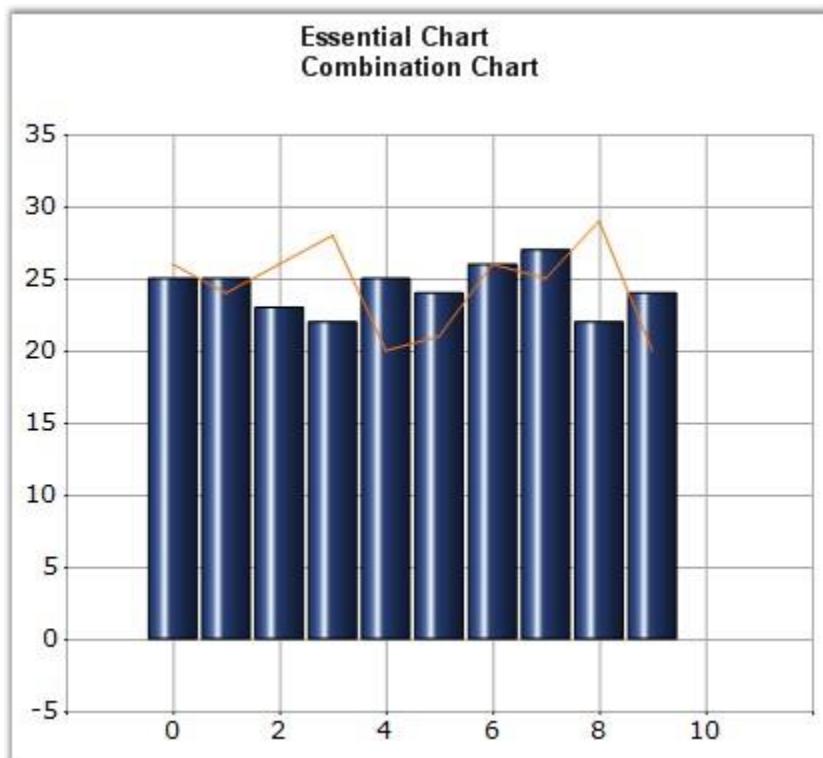
Border, DisplayText, DrawSeriesNameInDepth, ElementBorders, ImageIndex, Images, LightAngle, LightColor, PhongAlpha, Radar Type, RadarStyle  
Rotate, ShadingMode, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels,  
Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.10 Combination Chart

Combination Charts refers to the ability to display multiple data series in the same chart with each series visualized using different chart types. In Essential Chart, Chart types that are compatible with each other may be combined in the same Chart Area.

Typically it is a combination of a Line chart and a Column chart, sharing a common x-axis but with separate y-axes, one on either side of the chart.

One can change an existing chart to a combination chart by selecting the data series you want to change and then changing the chart type for that series.



*Figure 86: Chart displaying Line and Column Chart*

### Chart Details

Details	
<b>Number of Series</b>	One or more.
<b>Cannot be Combined with</b>	Pie, Bar, Polar, Radar.

Combination series can be added to the chart using the following code.

#### [C#]

```
ChartSeries series = this.chartControl1.Model.NewSeries ("Series Name",ChartSeriesType.Line);
series.Points.Add (0, 2);
series.Points.Add (1, 1);
series.Points.Add (2, 1);

// Create chart series and add data points into it.
ChartSeries series2 = this.chartControl1.Model.NewSeries ("Series Name",ChartSeriesType.Column);
series2.Points.Add (0, 1);
series2.Points.Add (1, 3);
series2.Points.Add (2, 4);

// Add the series to the chart series collection.
this.chartControl1.Series.Add (series);
this.chartControl1.Series.Add (series2);
```

#### [VB .NET]

```
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries ("Series Name", ChartSeriesType.Line)
series.Points.Add (0, 2)
series.Points.Add (1, 1)
series.Points.Add (2, 1)

' Create chart series and add data points into it.
Dim series2 As ChartSeries = Me.chartControl1.Model.NewSeries ("Series Name", ChartSeriesType.Column)
series2.Points.Add (0, 1)
series2.Points.Add (1, 3)
```

```
series2.Points.Add (2, 4)

' Add the series to the chart series collection.
Me.chartControll1.Series.Add (series)
Me.chartControll1.Series.Add (series2)
```

### Customization Options

Border, DisplayShadow, DisplayText, DrawColumnSeparatingLines, ElementBorders, ImageIndex, Images, LightAngle, LightColor, PhongAlpha, Spacing Between Series, ShadowInterior, ShadowOffset, FancyToolTip, Font, Interior, LegendItem, Name, PointsToolTipFormat, SmartLabels, Summary, Text, TextColor, TextFormat, TextOffset, TextOrientation, Visible

#### 4.4.11 Heat Map Charts

A heat map chart is a graphical representation of data where the values taken by a variable in two-dimensional map are represented as colors.



Figure 87: Chart displaying Heat Map Series

#### Chart Details

Details
---------

<b>Number of Y values per point</b>	2
<b>Number of Series</b>	One.
<b>Cannot be Combined with</b>	Any other chart types.

Combination series can be added to the chart using the following code.

**[C#]**

```
ChartSeries Stocks = new ChartSeries("Stocks",
ChartSeriesType.HeatMap);
Stocks.Points.Add(7,4, 10000);
Stocks.Points.Add(6,3, 5541);
Stocks.Points.Add(5,2, 6007);
Stocks.Points.Add(4,2, 5022);
Stocks.Points.Add(3,2.5, 6882);
Stocks.Points.Add(2,1.5, 6584);
Stocks.Points.Add(1,1, 2799);
this.chartControll1.Series.Add(Stocks);
```

**[VB .NET]**

```
Dim Stocks As ChartSeries = Me.chartControll1.Model.NewSeries ("Stocks",
ChartSeriesType.Line)
Stocks.Points.Add(7,4, 10000)
Stocks.Points.Add(6,3, 5541)
Stocks.Points.Add(5,2, 6007)
Stocks.Points.Add(4,2, 5022)
Stocks.Points.Add(3,2.5, 6882)
Stocks.Points.Add(2,1.5, 6584)
Stocks.Points.Add(1,1, 2799)

' Add the series to the chart series collection.
Me.chartControll1.Series.Add (Stocks)
```

## Features

The following table lists the properties of heat map chart with descriptions.

Property	Description
HeatMapStyle	Specifies styles of heat maps. The types are Rectangular, Vertical and Horizontal styles.

DisplayColorSwatch	Enables the color swatch of the heat map.
DisplayTitle	Enables or disables the series title in the left corner of the swatch.
StartText	Sets the text for the left label in the color swatch.
EndText	Sets the text for the right label in the color swatch.
LowestValueColor	Sets the lowest value color of the heat map chart.
HighestValueColor	Sets the highest value color of the heat map chart.
MiddleValueColor	Sets the middle value color of the heat map chart.
LabelMargin	Sets the margin for the left and right side labels.

**[C#]**

```
//Sets the Heat map style.
this.chartControl1.Series[0].ConfigItems.HeatMapItem.HeatMapStyle =
ChartHeatMapLayoutStyle.Rectangular;
//Display color swatch.
this.chartControl1.Series[0].ConfigItems.HeatMapItem.DisplayColorSwatch =
true;
//Sets the Series Title.
this.chartControl1.Series[0].ConfigItems.HeatMapItem.DisplayTitle =
true;
//Sets the left and right label text.
this.chartControl1.Series[0].ConfigItems.HeatMapItem.StartText = "US";
this.chartControl1.Series[0].ConfigItems.HeatMapItem.EndText = "Utah";
//Sets the lowest, highest and middle value color.
this.chartControl1.Series[0].ConfigItems.HeatMapItem.LowestValueColor =
Color.Red;
this.chartControl1.Series[0].ConfigItems.HeatMapItem.HighestValueColor =
Color.Blue;
this.chartControl1.Series[0].ConfigItems.HeatMapItem.MiddleValueColor =
Color.Yellow;
//Sets the value for label margin.
this.chartControl1.Series[0].ConfigItems.HeatMapItem.LabelMargins = 15;
```

**[VB .NET]**

```
'Sets the Heat map style.
Me.chartControl1.Series(0).ConfigItems.HeatMapItem.HeatMapStyle =
ChartHeatMapLayoutStyle.Rectangular
```

```
'Display color swatch.  
Me.chartControll1.Series(0).ConfigItems.HeatMapItem.DisplayColorSwatch =  
True  
'Sets the display title.  
Me.chartControll1.Series(0).ConfigItems.HeatMapItem.DisplayTitle = True  
'Sets the start and end text.  
series.ConfigItems.HeatMapItem.StartText = "US"  
series.ConfigItems.HeatMapItem.EndText = "Utah"  
'Sets the lowest, highest and middle value color.  
series.ConfigItems.HeatMapItem.LowestValueColor = Color.FromArgb(255,  
23, 0)  
series.ConfigItems.HeatMapItem.HighestValueColor = Color.FromArgb(81,  
168, 0)  
series.ConfigItems.HeatMapItem.MiddleValueColor = Color.Gold  
'Sets the margin for the left and right labels.  
series.ConfigItems.HeatMapItem.LabelMargins = 15
```

#### 4.4.12 Stacking Charts

Stacking Charts are similar to regular charts except that the y values stack on top of each other in the specified series order. Stacking charts help visualize data that is a sum of parts, each of which is in a series.

There are different types of stacking charts:

- [Stacking Area Chart](#)
- [Stacking Bar Chart](#)
- [Stacking Column Chart](#)
- [StackedArea100 Chart](#)
- [StackedBar100 Chart](#)
- [StackedColumn100 Chart](#)

#### 4.4.13 Step Charts

Step Charts are similar to regular charts except that the values are drawn continuously, step-by-step without any gaps in-between.

There are two types of step charts.

- [Step Area Chart](#)
- [Step Line Chart](#)

#### 4.4.14 Sparkline

##### Sparkline Overview

A Sparkline control is a type of information graphic characterized by its small size, high data density and lightweight. It presents trends and variations in a very condensed fashion. The Sparkline does not contain an axis scale and is intended to give a high level overview of what happened to the data over time.

##### Use Case Scenarios

A sparkline can display a trend based on adjacent data in a clear and compact graphical representation. The purpose of sparkline is to quickly see the data range difference with high density data and it is represented in lightweight graphical representation. You can use it as per your requirement.

The following screenshot shows three types of sparklines, which are drawn inside the grid control cell, based on row values.

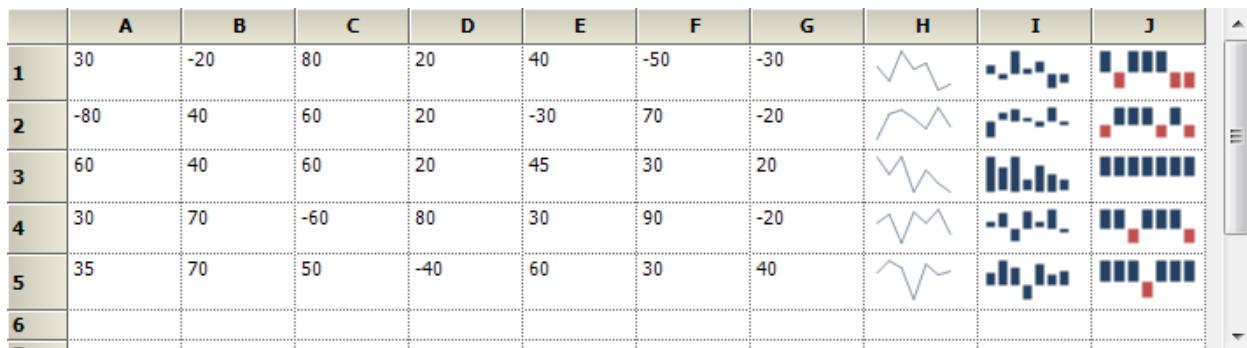


Figure 88: Sparkline control in RealTime

##### Tables for Properties, Methods, and Events

##### Properties

Property	Description	Type	Data Type	Reference links

Property	Description	Type	Data Type	Reference links
Type	<p>Specifies the types of spark lines.</p> <ul style="list-style-type: none"> <li>• Line</li> <li>• Column</li> <li>• WinLoss</li> </ul> <p>By default, it is set to <b>Line</b> type.</p>	NA	NA	NA
Source	Gets or sets the data source for sparkline data points	NA	NA	NA
LineStyle	Customizes the styles of Line sparkline	NA	NA	NA
ColumnStyle	Customizes the styles of Column and Winloss sparklines	NA	NA	NA
Markers	Enables the markers support to sparkline	NA	NA	NA
BackInterior	Customizes the background color of the control. By default, it is set to White color	NA	NA	NA

## Methods

Method	Description	Parameters	Type	Return Type	Reference links
GetHighPoint	Gets the highest point value from the sparkline	NA	NA	Void	NA
GetLowPoint	Gets the lowest point value from the sparkline	NA	NA	Void	NA

GetStartPoint	Gets the start point value from the sparkline	NA	NA	Void	NA
GetEndPoint	Gets the end point value from the sparkline	NA	NA	Void	NA

## Events

NA

## Sample Link

To access a Sparkline sample Demo:

1. Open the Syncfusion Dashboard.
2. Select User Interface.
3. Click the **Windows Forms** drop-down list and select **Explore Samples**.
4. Navigate to **Chart.Windows -> Samples -> 2.0 -> SparklineChart**.

## Types of Sparklines

Presently, Syncfusion SparkLine control supports three types of Sparklines and the sparkline control must be bound to a data source. It supports a variety of datasource such as DataTable and any component that implements the interface **IEnumerable**, **ICollection**, **IList**.

- Line
- Column
- WinLoss

## Drawing Line Sparkline in an Application

The line type of spark line represents a set of data points, connected by a line.

Refer to the following code snippets to draw the line sparkline.

**[C#.NET]**

```
//Set Sparkline points to source property
```

```
this.sparkLine1.Source =new double[] { 30, -20, 80, 20, 40, -50, -30,  
70, -40, 50 };  
  
//Set line type sparkline  
  
this.sparkLine1.Type = SparkLine.SparkLineType.Line;
```

**[VB.NET]**

```
'Set Sparkline points to source property  
  
Me.sparkLine1.Source = New Double() {30, -20, 80, 20, 40, -50,-30, 70,  
-40, 50}  
  
'Set line type sparkline  
  
Me.sparkLine1.Type = SparkLine.SparkLineType.Line
```



Figure 89: Line SparkLine

### Drawing Column Sparkline in an Application

The column type of spark line represents each data point by a column. The vertical column direction represents the negative or positive value.

Refer to the following code snippets to draw the column sparkline:

**[C#.NET]**

```
//Set Sparkline points to source property  
  
this.sparkLine1.Source =new double[] { 30, -20, 80, 20, 40, -50, -30,  
70, -40, 50 };  
  
//Set line type sparkline
```

```
this.sparkLine1.Type = SparkLine.SparkLineType.Column;
```

**[VB.NET]**

```
'Set Sparkline points to source property  
Me.sparkLine1.Source = New Double() {30, -20, 80, 20, 40, -50,-30, 70,  
-40, 50}  
  
'Set line type sparkline  
Me.sparkLine1.Type = SparkLine.SparkLineType. Column
```



Figure 90: Column SparkLine

### Drawing WinLoss Sparkline in an Application

The Winloss type of spark line is similar to column type but all columns have equal length for data points. The vertical column direction represents the negative or positive value.

Refer to the following code snippets to draw the WinLoss sparkline:

**[C#.NET]**

```
//Set Sparkline points to source property  
this.sparkLine1.Source =new double[] { 30, -20, 80, 20, 40, -50, -30,  
70, -40, 50 };  
  
//Set line type sparkline  
this.sparkLine1.Type = SparkLine.SparkLineType.WinLoss;
```

**[VB.NET]**

```
'Set Sparkline points to source property  
Me.sparkLine1.Source = New Double() {30, -20, 80, 20, 40, -50,-30, 70,  
-40, 50}  
  
'Set line type sparkline  
Me.sparkLine1.Type = SparkLine.SparkLineType.WinLoss
```



Figure 91: WinLoss SparkLine

### Marker Support

The markers are visual indicators to represent the location of data points in the Sparkline graph. The markers can support three types of sparklines.

Marker Property	Description
ShowMarker	Indicates whether the marker should be displayed at every data point location in line sparkline. By default it is set to False.
ShowHighPoint	Enables markers to show the highest values in all types of sparklines. By default it is set to False.
ShowLowPoint	Enables markers to show the lowest values in all types of sparklines. By default it is set to False.
ShowStartPoint	Enables markers show start values in all types of sparklines. By default it is set to False.
ShowEndPoint	Enables markers to show end values in all types of sparklines. By default it is set to False.

ShowNegativePoint	Enables markers to show negative values in all types of sparklines. By default it is set to False.
MarkerColor	Gets or sets the marker color for line type sparkline. This property color is set to sparkline marker when enabling the ShowMarker property.
HighPointColor	Gets or sets the high point color for line type sparkline. This property color is set to sparkline marker when enabling the ShowHighPoint property.
LowPointColor	Gets or sets the low point color to line type sparkline. This property color is set to sparkline marker when enabling the ShowLowPoint property.
StartPointColor	Gets or sets the start point color for line type sparkline. This property color is set to sparkline marker when enabling the ShowStartPoint property.
EndPointColor	Gets or sets the end point color to line type sparkline. This property color is set to sparkline marker when enabling the ShowEndPoint property.
NegativePointColor	Gets or sets the negative point color to line type sparkline. This property color is set to sparkline marker when enabling the ShowNegativePoint property.

### Markers Support for Line

This marker feature supports data points of line sparkline. You can choose the marker color for data points.

Refer to the following code snippets to enable the marker in line sparkline.

#### [C#.NET]

```
//To enable marker to sparkline for all data points
this.sparkLine1.Markers.ShowMarker =true;
```

#### [VB.NET]

```
'To enable marker to sparkline for all data points
Me.sparkLine1.Markers.ShowMarker =True
```

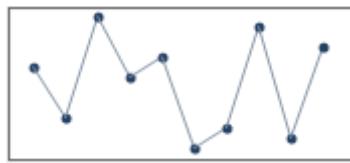


Figure 92: Marker for Line SparkLine

### Markers Support for Column

This marker feature supports High Points, Low Points, Start Point and Negative Point of column sparkline. You can choose the marker color for data points.

Refer to the following code snippets to enable the marker in column sparkline.

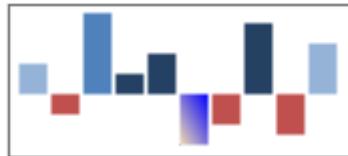
#### [C#.NET]

```
//To enable marker to sparkline high,low,start,end,negative data points  
this.sparkLine1.Markers.ShowHighPoint = true;  
this.sparkLine1.Markers.ShowLowPoint = true;  
this.sparkLine1.Markers.ShowStartPoint = true;  
this.sparkLine1.Markers.ShowEndPoint = true;  
this.sparkLine1.Markers.ShowNegativePoint= true;  
  
//To customize the marker color to low points  
this.sparkLine1.Markers.LowPointColor = new  
BrushInfo(GradientStyle.BackwardDiagonal, Color.Blue, Color.Wheat);
```

#### [VB.NET]

```
//To enable marker to sparkline high,low,start,end,negative data points  
Me.sparkLine1.Markers.ShowHighPoint = True  
Me.sparkLine1.Markers.ShowLowPoint = True  
Me.sparkLine1.Markers.ShowStartPoint = True  
Me.sparkLine1.Markers.ShowEndPoint = True  
Me.sparkLine1.Markers.ShowNegativePoint= True
```

```
//To customize the marker color to low points  
Me.sparkLine1.Markers.LowPointColor = new  
BrushInfo(GradientStyle.BackwardDiagonal, Color.Blue, Color.Wheat)
```



*Figure 93: Markers for Column SparkLine*

### **Markers Support for WinLoss**

This marker feature supports High Points, Low Points, Start Point and Negative Point of WinLoss Sparkline. The markers feature of WinLoss is the same as Column markers. You can choose the marker color for data points.

Refer to the following code snippets to enable the marker in column sparkline.

#### **[C#.NET]**

```
//To enable marker to sparkline high,low,start,end,negative data points  
this.sparkLine1.Markers.ShowHighPoint = true;  
this.sparkLine1.Markers.ShowLowPoint = true;  
this.sparkLine1.Markers.ShowStartPoint = true;  
this.sparkLine1.Markers.ShowEndPoint = true;  
this.sparkLine1.Markers.ShowNegativePoint= true;  
  
//To customize the marker color to low points  
this.sparkLine1.Markers.LowPointColor = new  
BrushInfo(GradientStyle.BackwardDiagonal, Color.Blue, Color.Wheat);
```

#### **[VB.NET]**

```
//To enable marker to sparkline high,low,start,end,negative data points
```

```
Me.sparkLine1.Markers.ShowHighPoint = True  
Me.sparkLine1.Markers.ShowLowPoint = True  
Me.sparkLine1.Markers.ShowStartPoint = True  
Me.sparkLine1.Markers.ShowEndPoint = True  
Me.sparkLine1.Markers.ShowNegativePoint= True  
  
//To customize the marker color to low points  
Me.sparkLine1.Markers.LowPointColor = new  
BrushInfo(GradientStyle.BackwardDiagonal, Color.Blue, Color.Wheat)
```

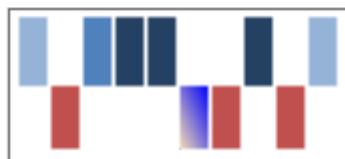


Figure 94: Markers for WinLoss SparkLine

## 4.5 Chart Series

Provide data for the chart through the **ChartSeries**. ChartSeries acts as a wrapper around data that is to be displayed, and associates styles with the data. The data that is to be displayed is contained in either the **IChartSeriesModel** or the **IEditableChartSeriesModel** implementation. The style used to display the points is stored in a contained implementation of **IChartSeriesStylesModel**.

Here is some sample code to create a new series and add it to the chart.

```
[C#]  
  
// 1) One way to create a series:  
ChartSeries series = new ChartSeries("Sales Performance",  
ChartSeriesType.Bar);  
series.Points.Add(0,200);  
series.Points.Add(1,300);  
// Remember to add the series to the chart.  
chartControl1.Series.Add(series);
```

```
// 2) Another way to create a series  
// This will automatically add the series to the chart  
ChartSeries series = chartControl1.Model.NewSeries("Sales Performance",  
ChartSeriesType.Bar);  
series.Points.Add(0,200);  
series.Points.Add(1,300);
```

**[VB.NET]**

```
' 1) One way to create a series:  
Dim series1 As ChartSeries = New ChartSeries("Sales Performance",  
ChartSeriesType.Bar)  
series.Points.Add(0,200)  
series.Points.Add(1,300)  
' Remember to add the series to the chart.  
Me.chartControl1.Series.Add(series)  
  
' 2) Another way to create a series  
' This will automatically add the series to the chart  
ChartSeries series = Me.chartControl1.Model.NewSeries("Sales  
Performance", ChartSeriesType.Bar)  
series.Points.Add(0,200)  
series.Points.Add(1,300)
```



**Note:** Same `ChartSeries` object being added to more than one chart is not supported. It binds the series to the default primary axis always.

## Chart Points

The `ChartPoint` class holds value information about a single point in a series (x and y values). The following table describes the kind of x and y values you can specify via a chart point.

Axis	Number of Values	Value Types
X	1	double or DateTime
Y	1 or more	double or DateTime

Here is some sample code that shows adding data points to the `Points` collection. You could also optionally create a `ChartPoint` instance first and then add it to the `Points` collection.

**[C#]**

```
// Option 1: 1 X double value; 2 double Y values in a point  
series1.Points.Add(0, 2.5, 3.5);  
  
// Option 2: 1 X double value; 1 DateTime Y value  
series2.Points.Add(1, DateTime.Now);  
  
// Option 3: 1 X DateTime value; 1 double Y value  
series1.Points.Add(DateTime.Now, 5.3);
```

**[VB .NET]**

```
' Option 1: 1 X double value; 2 double Y values in a point  
series1.Points.Add(0, 2.5, 3.5)  
  
' Option 2: 1 X double value; 1 DateTime Y value  
series2.Points.Add(1, DateTime.Now)  
  
' Option 3: 1 X DateTime value; 1 double Y value  
series1.Points.Add(DateTime.Now, 5.3)
```

## ValueType

Always use the **ChartAxis.ValueType** property to specify what kind of values you have added in the series data points for the corresponding axis.

**[C#]**

```
// To specify DateTime values in the X axis  
this.chartControl1.PrimaryXAxis.ValueType = ChartValueType.DateTime;  
  
// To specify Double values  
this.chartControl1.PrimaryXAxis.ValueType = ChartValueType.Double;
```

**[VB .NET]**

```
' To specify DateTime values in the X axis  
Me.chartControl1.PrimaryXAxis.ValueType = ChartValueType.DateTime  
  
' To specify Double values  
Me.chartControl1.PrimaryXAxis.ValueType = ChartValueType.Double
```

**Note :** To display the text right next to the data points, the **DisplayText** property of the data point's style should be set.

The Chart Series features are elaborated in more detail in the following sub-topics.

### 4.5.1 Series Customization

Essential Chart offers numerous appearance and behavior customization capabilities at the series level and on individual points.

Some of these options are applicable only for the whole series while the rest could be applied on the specific data points. Similarly some of these options are specific to certain chart types.

Note that styles set at the series level automatically propagate to the points in the series.

Interestingly the Chart control lets the user to edit the styles of a series by double clicking on it during run-time. This feature can be turned on by setting the **AllowUserEditStyles** property to true.

The table below lists the customization options available in **ChartSeries** and their restrictions.

Customization Option	Applies to Series or DataPoints*	Applies to Chart Type
<a href="#">AngleOffset</a>	Series	Pie Chart
<a href="#">Border</a>	Series and points	Pyramid, Funnel, Area, Bar, Bubble, Column Chart, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Polar and Radar Chart and Pie Chart.
<a href="#">BubbleType</a>	Series	Bubble chart.
<a href="#">ColumnDrawMode</a>		Column Chart, ColumnRange Chart, Bar Chart,

		BoxAndWhisker Chart, Gantt Chart.
<a href="#">ColumnWidthMode</a>	Series	Column charts
<a href="#">ColumnFixedWidth</a>	Series	Column charts
<a href="#">ColumnType</a>	Series	Column
<a href="#">ColorsMode</a>	Series	Renko chart.
<a href="#">DarkLightPower</a>	Series	Renko chart.
<a href="#">DisplayShadow</a>	Series and points	Area Chart, Bar Chart, Bubble Chart, Column Chart, Stacking Column Chart, Stacking Column100 Chart, Line Chart, Spline Chart, Rotated Spline chart, Stepline Chart, Candle Chart, Kagi Chart, Point and Figure Chart, Renko Chart, Threeline Break Charts, Gantt Chart, Histogram chart, Tornado Chart, Combination Chart, Box and Whisker Chart.
<a href="#">DisplayText</a>	Series and points	All Chart types.
<a href="#">DoughnutCoeficient</a>	Series	Pie Chart.
<a href="#">DrawColumnSeparatingLines</a>	Series	Column Chart.
<a href="#">DrawErrorBars</a>	Series	Column Chart, Line Chart and HiLo Chart.
<a href="#">DrawHistogramNormalDistribution</a>	Series	Histogram chart.
<a href="#">DrawSeriesNameInDepth</a>	Series	All Chart types.
<a href="#">ElementBorders</a>	Series and points	Area Charts, Bar Charts, Bubble Chart, Column Charts, Line Charts, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Tornado Chart,

		Polar and Radar Chart.
<a href="#">EnablePhongStyle</a>	Series	Bubble Chart.
<a href="#">ErrorBarsSymbolShape</a>	Series	Column Chart, Line Chart and HiLo Chart.
<a href="#">ExplodedAll</a>	Series	Pie Chart, Doughnut Chart.
<a href="#">ExplodedIndex</a>	Series	Pie Chart.
<a href="#">ExplosionOffset</a>	Series	Pie Chart.
<a href="#">FancyToolTip</a>	Series	All Chart Types.
<a href="#">FigureBase</a>	Series	Funnel and Pyramid chart.
<a href="#">FillMode</a>	Series	Pie Chart
<a href="#">FunnelMode</a>	Series	Funnel and Pyramid chart.
<a href="#">Font</a>	Series and points	All Chart types.
<a href="#">GanttDrawMode</a>	Series	Gantt Chart.
<a href="#">GapRatio</a>	Series	Funnel and Pyramid chart.
<a href="#">Gradient</a>	Series	Pie Chart.
<a href="#">HeightBox</a>	Series	Point And Figure Chart.
<a href="#">HeightByAreaDepth</a>	Series	Pie Chart.
<a href="#">HeightCoeficient</a>	Series	Pie Chart.
<a href="#">HighlightInterior</a>	Series	Bar Charts, Pie, Funnel, Pyramid, Bubble, Column, Area, Stacking Area, Stacking Area100, Line Charts, Box and Whisker, Gantt Chart and Tornado Chart.
<a href="#">HitTestRadius</a>	Series	Line Chart and Step Line Chart.
<a href="#">ImageIndex</a>	Series and points	Area Charts, Bar Charts, Bubble Chart, Column Charts, Line Charts, Candle Chart,

		Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Tornado Chart, Polar and Radar Chart.
<a href="#">Images</a>	Series and points	Area Charts, Bar Charts, Bubble Chart, Column Charts, Line Charts, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Tornado Chart, Polar and Radar Chart.
<a href="#">InnerRadius</a>	Series	Pie Chart.
<a href="#">Interior</a>	Series and points	All Chart Types.
<a href="#">LabelPlacement</a>	Series	Funnel and Pyramid Charts.
<a href="#">LabelStyle</a>	Series	Funnel and Pyramid, Pie
<a href="#">LegendItem</a>	Series	All Chart Types.
<a href="#">LightAngle</a>	Series	Scatter Chart, Column Charts , Bar Charts, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Polar and Radar Chart.
<a href="#">LightColor</a>	Series	Scatter Chart, Column Charts , Bar Charts, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Polar and Radar Chart.
<a href="#">Name</a>	Series	All chart types.
<a href="#">NumberOfHistogramIntervals</a>	Series	Histogram Chart.
<a href="#">OpenCloseDrawMode</a>	Series	HiLo OpenClose chart.
<a href="#">OptimizePiePointPositions</a>	Series	Pie chart
<a href="#">PhongAlpha</a>	Series	Column Chart.
<a href="#">PieType</a>	Series	Pie chart

<a href="#">PieWithSameRadius</a>	Series	Pie chart and Doughnut chart.
<a href="#">PointsToolTipFormat</a>	Series	All Chart Types.
<a href="#">PointWidth</a>	Series and points	Gantt Chart.
<a href="#">PriceDownColor</a>	Series	Financial types
<a href="#">PriceUpColor</a>	Series	Financial types
<a href="#">PyramidMode</a>	Series	Pyramid
<a href="#">Radar Type</a>	Series	Polar and Radar Chart.
<a href="#">RadarStyle</a>	Series	Polar and Radar Chart.
<a href="#">RelatedPoints</a>	Series and points	Gantt Chart.
<a href="#">ReversalAmount</a>	Series	Kagi, PointAndFigure, Renko
<a href="#">Rotate</a>	Series	Column Charts, Bar Charts, Area charts, Line Chart, Spline Chart, Stepline Chart, Candle Chart, HiLo Chart, HiLo Open Chart, Kagi Chart, BoxandWhisker chart, Histogram chart, Polar and Radar Chart.
<a href="#">ScatterConnectType</a>	Series	Scatter Chart.
<a href="#">ScatterSplineTension</a>	Series	Scatter Chart.
<a href="#">SeriesToolTipFormat</a>	Series	Area Charts.
<a href="#">ShadingMode</a>	Series	Column Chart, BarCharts, Candle Chart, HiLO Chart, HiLoOpenClose Chart, Tornado chart, BoxandWhisker chart, Gantt Chart, Histogram Chart, Polar and Radar Chart.
<a href="#">ShadowInterior</a>	Series and points	Column Charts, Bubble Chart, Line Charts, BarCharts, Candle Chart, Kagi Chart, Point and Figure Chart, Renko Chart,

		Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Pie Chart, Polar and Radar Chart.
<a href="#">ShadowOffset</a>	Series and points	Column Charts, Bubble Chart, Line Charts, BarCharts, Candle Chart, Kagi Chart, Point and Figure Chart, Renko Chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Pie Chart, Polar and Radar Chart.
<a href="#">ShowDataBindLabels</a>	Series	Pie Chart, Doughnut Chart, Funnel and Pyramid charts.
<a href="#">ShowHistogramDataPoints</a>	Series	Histogram Chart
<a href="#">ShowTicks</a>	Series	Pie Chart.
<a href="#">SmartLabels</a>	Series	All Chart Types.
<a href="#">Spacing</a>	Series and Points.	Column Charts, BarCharts, Box and Whisker Chart, Gantt Chart, Tornado Chart.
<a href="#">Spacing Between Series</a>	Series	Area Charts, BarCharts, Line Charts, Bubble Chart, Financial Charts, Gantt Chart, Histogram chart, Tornado Chart, Combination Chart, Box and Whisker Chart.
<a href="#">SpacingBetweenPoints</a>	Series Points	Column Chart, Bar Chart, HiLo Chart, HiLo Open Close Chart, Candle Chart, Tornado Chart, Boxes and Whisker Chart.
<a href="#">StepItem.Inverted</a>	Series	StepAreaChart, StepLine Chart.
<a href="#">Summary</a>	Series	All Chart Types.

<a href="#">Symbol</a>	Series and points	Column Chart, Bar Chart, Bubble Chart, Financial Chart, Line Chart, BoxandWhisker Chart, Gantt chart, Tornado chart, Radar Chart
<a href="#">Text (Series)</a>	Series	All Chart Types.
<a href="#">Text (Style)</a>	Series and Points	All Chart Types.
<a href="#">TextColor</a>	Series and points	All Chart Types.
<a href="#">TextFormat</a>	Series and points	All Chart Types.
<a href="#">TextOffset</a>	Series and points	All Chart Types.
<a href="#">TextOrientation</a>	Series and points	All Chart Types.
<a href="#">ToolTip</a>	Series and points	Scatter Chart.
<a href="#">ToolTipFormat</a>	Series and points	Scatter Chart.
<a href="#">Visible</a>	Series	All Chart Types.
<a href="#">VisibleAllPies</a>	Series	Pie Chart.
<a href="#">XType</a>	Series	All Chart Types.
<a href="#">YType</a>	Series	All Chart Types.
<a href="#">ZOrder</a>	Series	Gantt chart, StackingBar chart, StackingBar100 chart, StackingColumn chart, StackedColumn100 chart, StackingArea chart, StackingArea100 chart.

\* Indicates whether the property affects ALL the points in the series or if the property can be set on individual points as well.

#### 4.5.1.1 AngleOffset

The offset angle that is to be used when rendering Pie charts.

Details	
<b>Possible Values</b>	Accepts real values like 45f, 90f etc.
<b>Default Value</b>	<b>0</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	PieChart

Here is some sample code.

**[C#]**

```
// Create chart series and add data points into it.  
ChartSeries series1 = this.chartControl1.Model.NewSeries("Market");  
series1.Points.Add(0, 20);  
series1.Points.Add(1, 28);  
series1.Type = ChartSeriesType.Pie;  
// Add the series to the chart series collection.  
this.chartControl1.Series.Add(series1);  
  
this.chartControl1.Series3D = true;  
this.chartControl1.Series[0].ConfigItems.PieItem.AngleOffset = 45f;
```

**[VB.NET]**

```
' Create chart series and add data points into it.  
Private series1 As ChartSeries =  
Me.chartControl1.Model.NewSeries("Market")  
series1.Points.Add(0, 20)  
series1.Points.Add(1, 28)  
series1.Type = ChartSeriesType.Pie  
' Add the series to the chart series collection.  
Me.chartControl1.Series.Add(series1)  
  
Me.chartControl1.Series3D = True  
Me.chartControl1.Series(0).ConfigItems.PieItem.AngleOffset = 45f
```

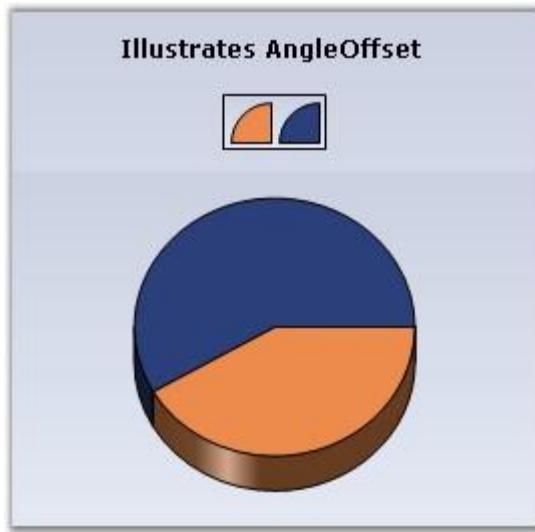


Figure 95: PieChart with No AngleOffset

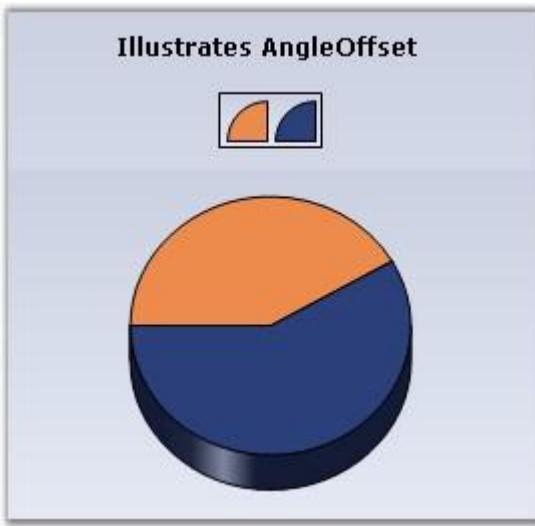


Figure 96: PieChart with AngleOffset = "45f"

#### See Also

[Pie Chart](#)

#### 4.5.1.2 Border

The user can also set the Border color and Border style for the chart series.

Details	
<b>Possible Values</b>	Any Color, Width, Style for the Borders
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>Color</b> - Black</li> <li>• <b>Width value</b> - 1</li> <li>• <b>DashStyle</b> - Solid</li> </ul>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	Pyramid, Funnel, Area, Bar, Bubble, Column Chart, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Polar and Radar Chart and Pie Chart

The line type can be configured using the **ChartSeries.Style.Border** property as in the following example.

**[C#]**

```
// Set the border style required for the column chart.
series.Style.Border.Width = 3;
series.Style.Border.Color = Color.White;

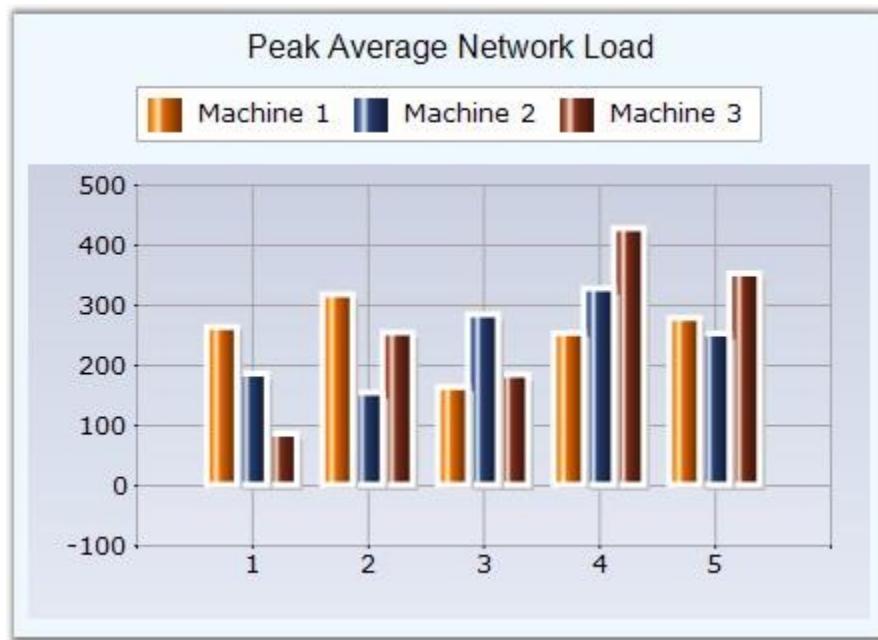
// Set the Series style
series.Style.DisplayShadow = true;
series.Style.ShadowInterior = new
Syncfusion.Drawing.BrushInfo(Color.White);
series.Style.ShadowOffset = new Size(3, 3);
```

**[VB .NET]**

```
' Set the border style required for the column chart.
series.Style.Border.Width = 3
series.Style.Border.Color = Color.White

' Set the Series style
series.Style.DisplayShadow = True
series.Style.ShadowInterior = New
Syncfusion.Drawing.BrushInfo(Color.White)
```

```
series.Style.ShadowOffset = New Size(3, 3)
```



**Figure 97: Border Lined Column Chart**

To apply this on specific data points:

**[C#]**

```
//Sets border for the 1st point in 1st series
series1.Styles[0].Border.Width = 3;
series1.Styles[0].Border.Color = Color.White;

//Sets border for the 3rd point in 2nd series
series2.Styles[2].Border.Width = 3;
series2.Styles[2].Border.Color = Color.White;
```

**[VB .NET]**

```
'Sets border for the 1st point in 1st series
series1.Styles(0).Border.Width = 3
series1.Styles(0).Border.Color = Color.White
```

```
'Sets border for the 3rd point in 2nd series
series2.Styles(2).Border.Width = 3
series2.Styles(2).Border.Color = Color.White
```

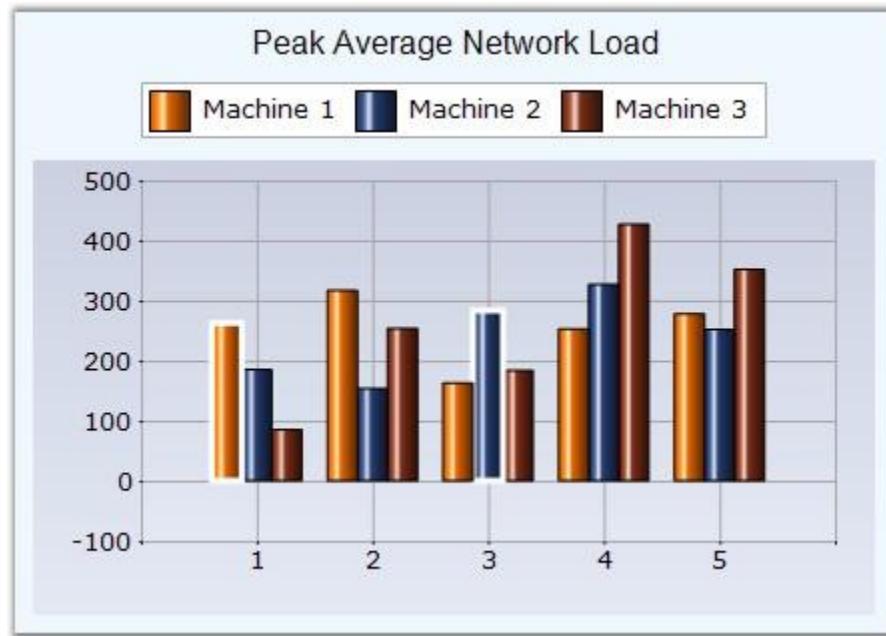


Figure 98: Individual Data Points with White Border

### See Also

[Pyramid Chart](#), [Funnel Chart](#), [Area Charts](#), [BarCharts](#), [Bubble Chart](#), [Column Chart](#), [Candle Chart](#), [Renko chart](#), [Three Line Break Chart](#), [Box and Whisker Chart](#), [Gantt Chart](#), [Histogram Chart](#), [Tornado Chart](#), [Polar and Radar Chart](#), [Pie Chart](#)

### 4.5.1.3 BubbleType

**BubbleType** - Specifies whether to render the data point symbols as circle, square or as image.

Details	
Possible Values	<ul style="list-style-type: none"> <li><b>Circle</b> - Symbol is rendered as a circle</li> <li><b>Square</b> - Symbol is rendered as a square</li> </ul>

	<ul style="list-style-type: none"><li>• <b>Image</b> - Symbol is rendered as an image</li></ul>
<b>Default Value</b>	<b>Circle</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Bubble

Here is some sample code to specify an Image BubbleType.

#### **Series wide setting**

##### **[C#]**

```
this.chartControl1.Series[0].ConfigItems.BubbleItem.BubbleType =  
ChartBubbleType.Image;  
this.chartControl1.Series[0].Style.Images = new  
ChartImageCollection(this.imageList1.Images );  
this.chartControl1.Series[0].Style.ImageIndex = 0;
```

##### **[VB .NET]**

```
Me.chartControl1.Series[0].ConfigItems.BubbleItem.BubbleType =  
ChartBubbleType.Image  
Me.chartControl1.Series[0].Style.Images = New  
ChartImageCollection(Me.imageList1.Images)  
Me.chartControl1.Series[0].Style.ImageIndex = 0
```

#### **Specific Data Point Setting**

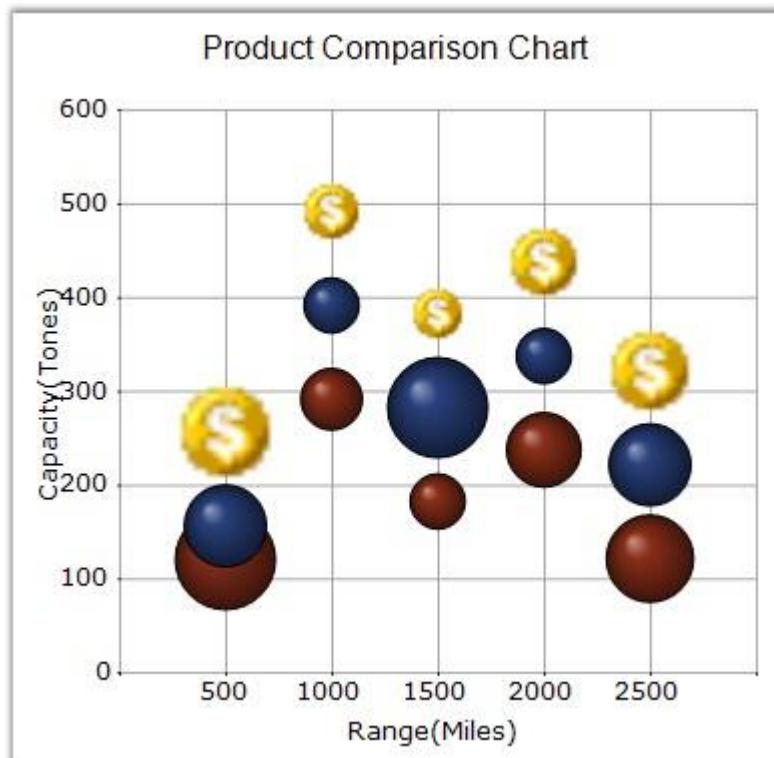
Specify image for specific data points.

##### **[C#]**

```
this.chartControl1.Series[0].Styles[0].Images = new  
ChartImageCollection(this.imageList1.Images );  
this.chartControl1.Series[0].Styles[0].ImageIndex = 0;  
this.chartControl1.Series[0].Styles[1].Images = new  
ChartImageCollection(this.imageList1.Images );  
this.chartControl1.Series[0].Styles[1].ImageIndex = 1;
```

**[VB.NET]**

```
Me.chartControl1.Series[0].Styles[0].Images = New  
ChartImageCollection(Me.imageList1.Images)  
Me.chartControl1.Series[0].Styles[0].ImageIndex = 0  
Me.chartControl1.Series[0].Styles[1].Images = New  
ChartImageCollection(Me.imageList1.Images)  
Me.chartControl1.Series[0].Styles[1].ImageIndex = 1
```



*Figure 99: Image BubbleType Chart Series*

**See Also**

[Bubble Chart](#)

#### 4.5.1.4 ColumnDrawMode

Indicates the drawing mode of columns in charts when there are multiple series.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>InDepthMode</b> - Columns from different series are drawn at different depths.</li> <li>• <b>PlaneMode</b> - Columns from all series are drawn side-by-side.</li> <li>• <b>ClusteredMode</b> - Columns from all series are drawn in depth with the same size.</li> </ul>
<b>Default Value</b>	<b>InDepthMode</b>
<b>2D / 3D Limitations</b>	3D only
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Column Chart, ColumnRange Chart, Bar Chart, BoxAndWhisker Chart, Gantt Chart

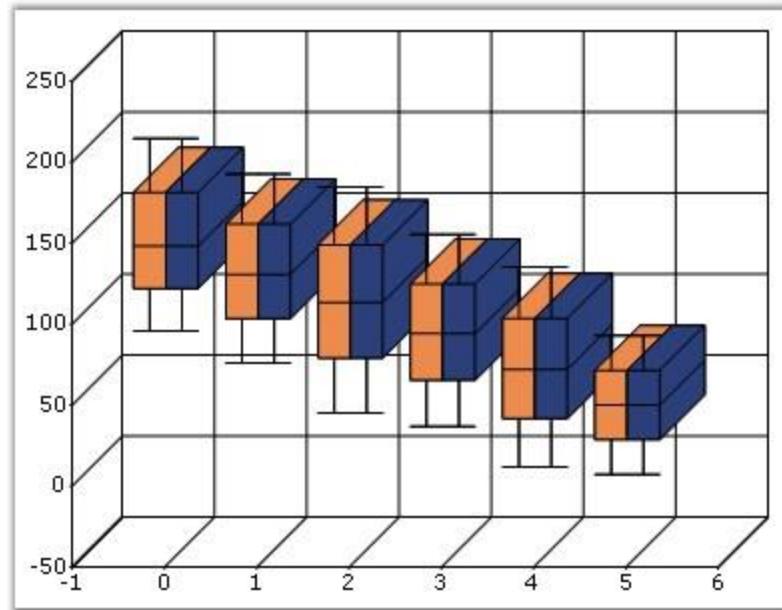
Here is the sample code snippet using **ColumnDrawMode** in Column Chart.

[C#]

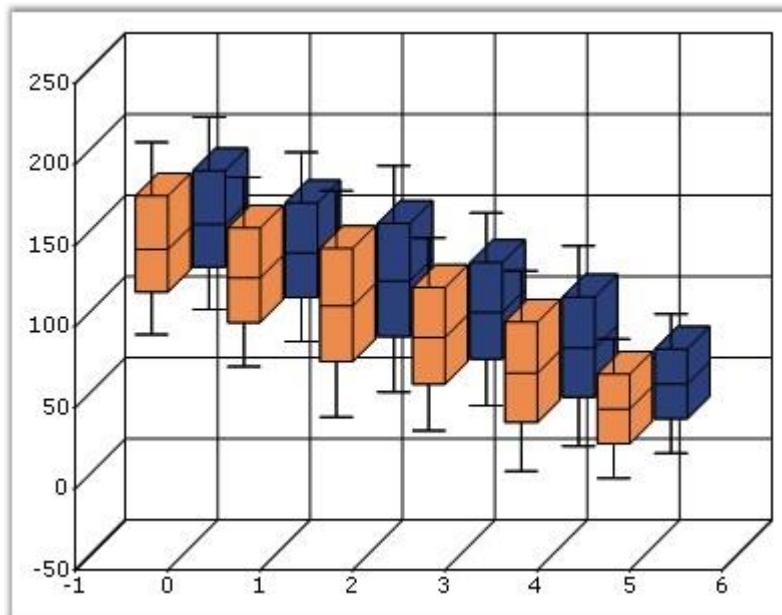
```
this.chartControl1.ColumnDrawMode = ChartColumnDrawMode.PlaneMode;
```

[VB .NET]

```
Me.chartControl1.ColumnDrawMode = ChartColumnDrawMode.PlaneMode
```



*Figure 100: ColumnDrawMode set to "PlaneMode"*



*Figure 101: ColumnDrawMode set to "InDepthMode"*

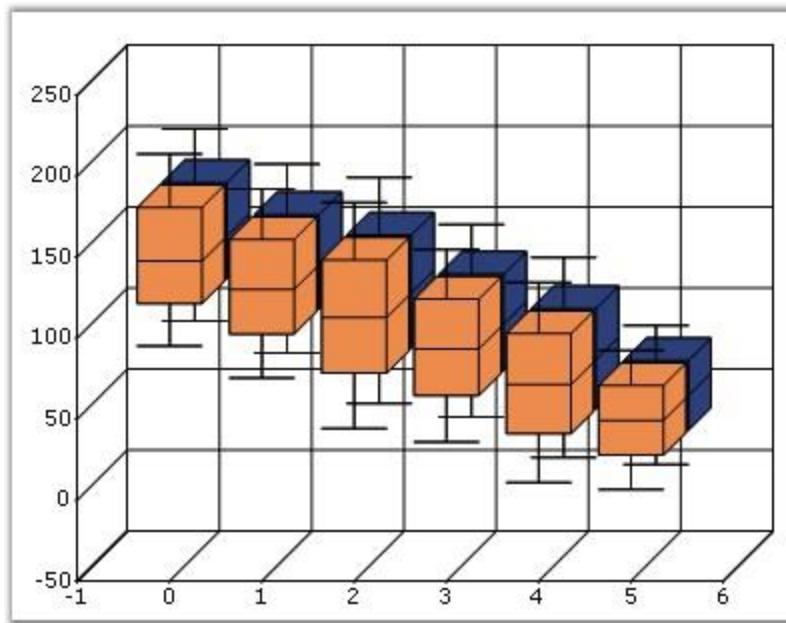


Figure 102: ColumnDrawMode set to "ClusteredMode"

## See Also

[Column Chart](#), [ColumnRange Chart](#), [Bar Chart](#), [BoxAndWhisker Chart](#), [Gantt Chart](#)

### 4.5.1.5 ColumnWidthMode

Specifies the width drawing mode for the columns in a column chart.

Details	
Possible Values	
	<ul style="list-style-type: none"> <li><b>DefaultWidthMode</b> - The width of the columns will always be calculated to fill the space between columns.</li> <li><b>FixedWidthMode</b> - The width should be given in Series.Points[i].YValues[1], in pixels. If the width of the columns are not given in point YValues[1], then they are calculated to fill the space between columns.</li> <li><b>RelativeWidthMode</b> - Similar to the</li> </ul>

	FixedWidthMode, the width is specified in YValues[1], but in units of X-axis range.
<b>Default Value</b>	<b>DefaultWidthMode</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Column charts, BoxAndWhiskerChart, Candle Chart

Here is some sample code.

**[C#]**

```
ChartSeries series1 = new ChartSeries("Series");
series.Points.Add(1, 24);
series.Points.Add(2, 36);
series.Points.Add(3, 48);
chartControl1.Series.Add(series1);
chartControl1.ColumnWidthMode = ChartColumnWidthMode.DefaultWidthMode;
```

**[VB .NET]**

```
Dim series1 As ChartSeries = New ChartSeries("Series")
series.Points.Add(1, 24)
series.Points.Add(2, 36)
series.Points.Add(3, 48)
chartControl1.Series.Add(series1)
chartControl1.ColumnWidthMode = ChartColumnWidthMode.DefaultWidthMode
```



*Figure 103: Column Chart with DefaultWidthMode*

**[C#]**

```
double Interval = this.chartControl1.PrimaryXAxis.Range.Interval;

ChartSeries series = this.chartControl1.Model.NewSeries("Team 1");
// 2nd Y value specifies the column width
series.Points.Add(1, new double[] { 24, Interval * 0.75 });
series.Points.Add(2, new double[] { 36, Interval * 0.75 });
series.Points.Add(3, new double[] { 48, Interval * 0.75 });
this.chartControl1.Series.Add(series);

this.chartControl1.ColumnWidthMode =
ChartColumnWidthMode.RelativeWidthMode;
```

**[VB .NET]**

```
Dim Interval As Double = Me.chartControl1.PrimaryXAxis.Range.Interval

Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Team 1")
' 2nd Y value specifies the column width
```

```
series.Points.Add(1, New Double() { 24, Interval * 0.75 })
series.Points.Add(2, New Double() { 36, Interval * 0.75 })
series.Points.Add(3, New Double() { 48, Interval * 0.75 })
Me.chartControl1.Series.Add(series)

Me.chartControl1.ColumnWidthMode =
ChartColumnWidthMode.RelativeWidthMode
```

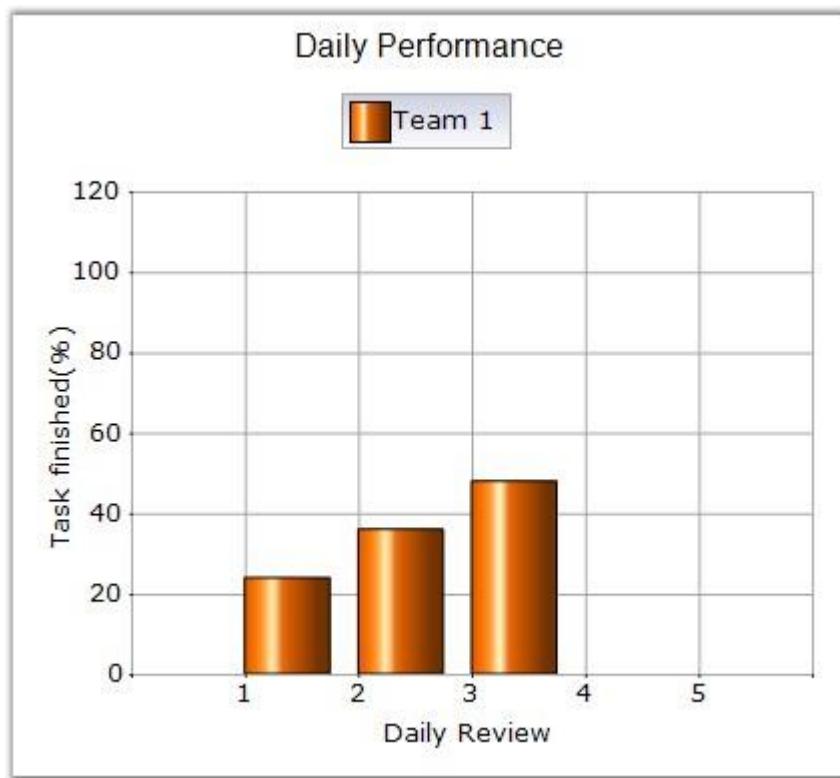


Figure 104: Column Chart with RelativeWidthMode

[C#]

```
ChartSeries series1 = new ChartSeries("Series");
// 2nd Y value specifies the column width
series1.Points.Add(1, new double[] { 24, 25 });
series1.Points.Add(2, new double[] { 36, 25 });
series1.Points.Add(3, new double[] { 48, 25 });
chartControl1.Series.Add(series1);
chartControl1.ColumnWidthMode = ChartColumnWidthMode.FixedWidthMode;
```

[VB.NET]

```
Dim series1 As ChartSeries = New ChartSeries("Series")
' 2nd Y value specifies the column width
series1.Points.Add(1, New Double() { 24, 25})
series1.Points.Add(2, New Double() { 36, 25})
series1.Points.Add(3, New Double() { 48, 25})
chartControl1.Series.Add(series1)
chartControl1.ColumnWidthMode = ChartColumnWidthMode.FixedWidthMode
```



**Note:** The width of the column can also be specified by **ColumnFixedWidth** property. If both second Y value and **ColumnFixedWidth** are specified, second Y value takes higher priority.

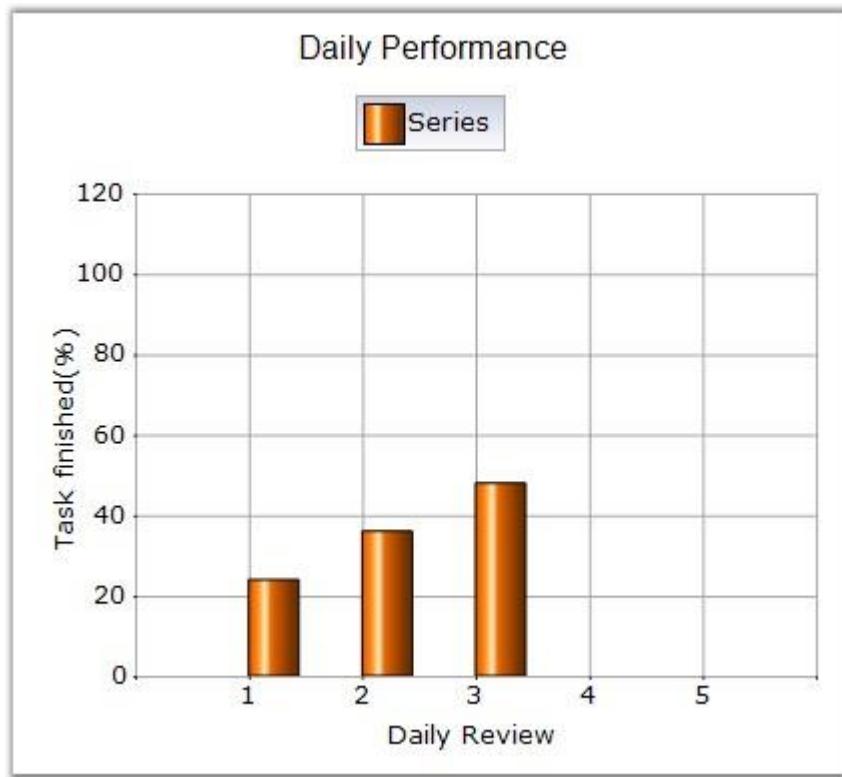


Figure 105: Column Chart with FixedWidthMode

#### See Also

[Column charts](#), [BoxAndWhiskerChart](#), [Candle Chart](#), [ColumnFixedWidth](#)

#### 4.5.1.6 ColumnFixedWidth

Specifies the width of each column when [ColumnWidthMode](#) is set to FixedWidthMode.

Details	
<b>Possible Values</b>	An integer value
<b>Default Value</b>	<b>20</b>
<b>2D / 3D Limitations</b>	None
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Column Charts, BoxAndWhiskerChart, Candle Chart

Here is some sample code.

#### [C#]

```
ChartSeries series1 = new ChartSeries("Series");
series1.Points.Add(1, new double[] { 24 });
series1.Points.Add(2, new double[] { 36 });
series1.Points.Add(3, new double[] { 48 });
chartControl1.Series.Add(series1);
chartControl1.ColumnWidthMode = ChartColumnWidthMode.FixedWidthMode;
chartControl1.ColumnFixedWidth = 45;
```

#### [VB .NET]

```
Dim series1 As ChartSeries = New ChartSeries("Series")
series1.Points.Add(1, New Double() { 24 })
series1.Points.Add(2, New Double() { 36 })
series1.Points.Add(3, New Double() { 48 })
chartControl1.Series.Add(series1)
chartControl1.ColumnWidthMode = ChartColumnWidthMode.FixedWidthMode
chartControl1.ColumnFixedWidth = 45
```



**Note:** The ColumnFixedWidth property can be overridden by specifying a second y value in the data point. See [ColumnWidthMode](#) for a sample.

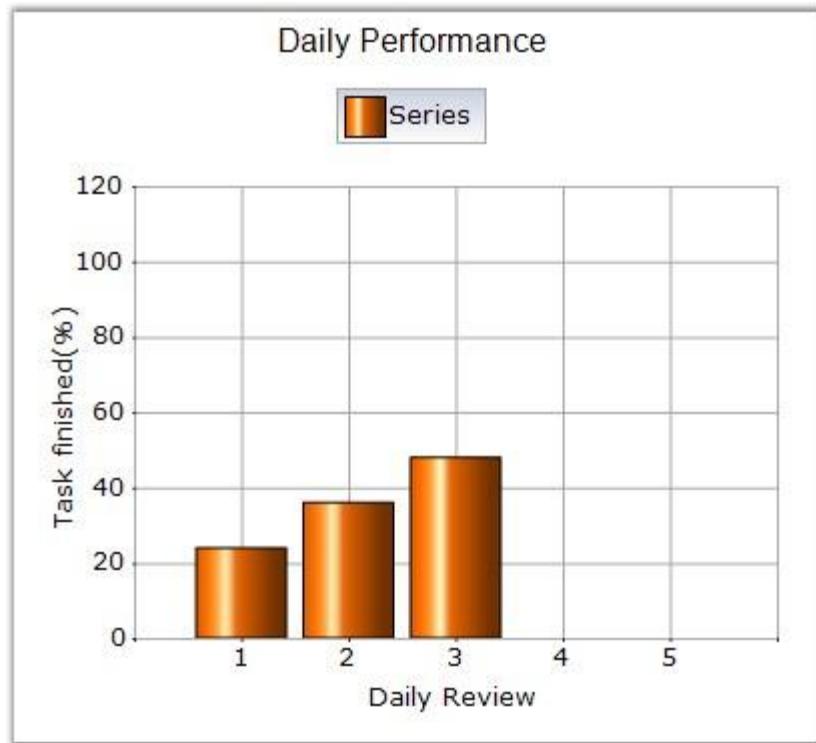


Figure 106: ColumnChart with ColumnFixedWidth = "45"

#### See Also

[Column charts](#), [BoxAndWhiskerChart](#), [ColumnWidthMode](#), [Candle Charts](#)

#### 4.5.1.7 ColumnType

Specifies whether the columns should be rendered as bars or cylinders.

Details	
<b>Possible Values</b>	Box - Renders the columns as boxes. Cylinder - Renders the columns as cylinders.
<b>Default Value</b>	Box
<b>2D / 3D Limitations</b>	3D only
<b>Applies to Chart Element</b>	All series

<b>Applies to Chart Types</b>	Column Chart, Column Range Chart, Stacking Column Chart, Candle Chart, Bar Chart, Stacking Bar Chart
-------------------------------	------------------------------------------------------------------------------------------------------

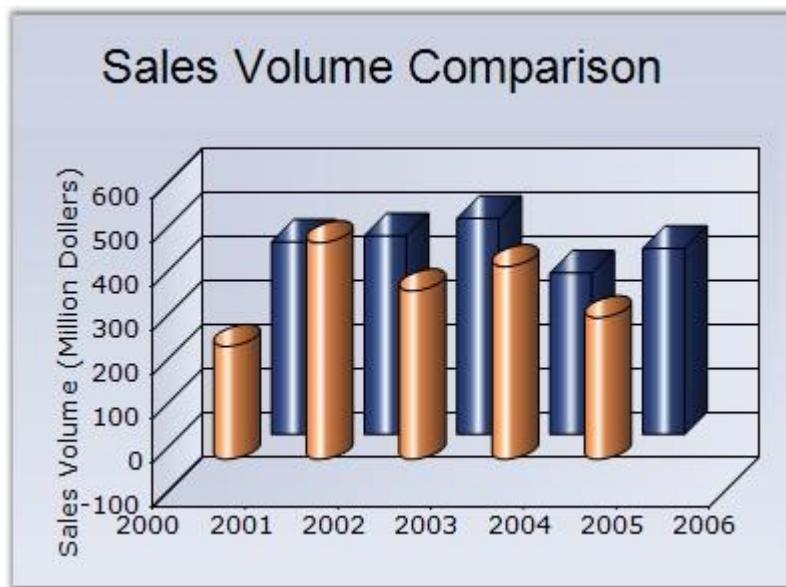
Here is some sample.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.ColumnItem.ColumnType =  
ChartColumnType.Cylinder;  
this.chartControl1.Series[1].ConfigItems.ColumnItem.ColumnType =  
ChartColumnType.Box;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.ColumnItem.ColumnType =  
ChartColumnType.Cylinder  
Me.chartControl1.Series(1).ConfigItems.ColumnItem.ColumnType =  
ChartColumnType.Box
```



*Figure 107: Column Chart*

**See Also**

[Column Chart](#), [Column Range Chart](#), [Stacking Column Chart](#), [Candle Chart](#), [Bar Chart](#), [Stacking Bar Chart](#)

#### 4.5.1.8 ColorsMode

Gets / sets ColorsMode of the boxes in the financial chart types.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>DarkLight</b> -Draws series data points as a darklight colorsmode.</li> <li>• <b>Fixed</b> - Draws series data points as a Fixed colorsmode.</li> <li>• <b>Mixed</b> - Draws series data points as a Mixed colorsmode.</li> </ul>
<b>Default Value</b>	Fixed
<b>2D / 3D Limitations</b>	None
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Renko Chart (Financial Chart)

Here is some sample code.

**[C#]**

```
// Setting ColorsMode for series
this.chartControl1.Series[0].ConfigItems.FinancialItem.ColorsMode =
ChartFinancialColorMode.DarkLight;
```

**[VB .NET]**

```
' Setting ColorsMode for series
Me.chartControl1.Series(0).ConfigItems.FinancialItem.ColorsMode =
```

`ChartFinancialColorMode.DarkLight`

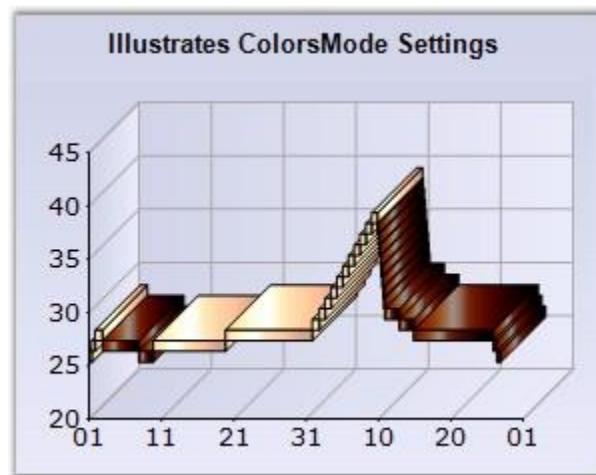


Figure 108: Renko Chart with "DarkLight" ColorsMode

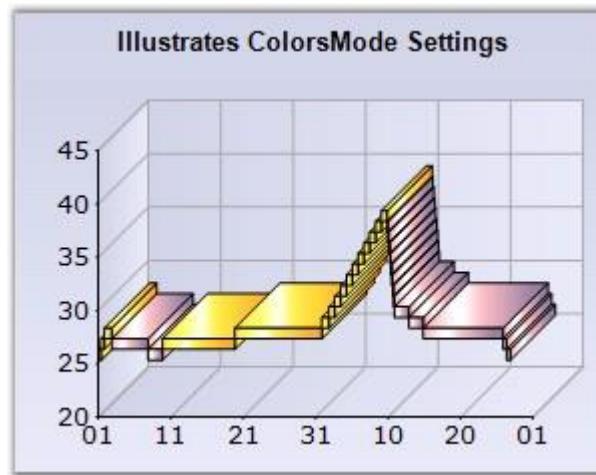


Figure 109: Renko Chart with "Mixed" ColorsMode

## See Also

[Renko Chart](#)

### 4.5.1.9 DarkLightPower

Gets or sets the intensity of the dark and light colors used in DarkLight color mode.

Details	
<b>Possible Values</b>	Ranges from 0 to 255 bytes
<b>Default Value</b>	100
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Renko Chart (Financial Charts)

Here is some sample code.

**[C#]**

```
// Setting ColorsMode as DarkLight
this.chartControl1.Series[0].ConfigItems.FinancialItem.ColorsMode =
ChartFinancialColorMode.DarkLight;
// Setting the power value of the darklight
this.chartControl1.Series[0].ConfigItems.FinancialItem.DarkLightPower =
200;
```

**[VB.NET]**

```
' Setting ColorsMode as DarkLight
Me.chartControl1.Series(0).ConfigItems.FinancialItem.ColorsMode =
ChartFinancialColorMode.DarkLight
' Setting the power value of the darklight
Me.chartControl1.Series(0).ConfigItems.FinancialItem.DarkLightPower =
200
```

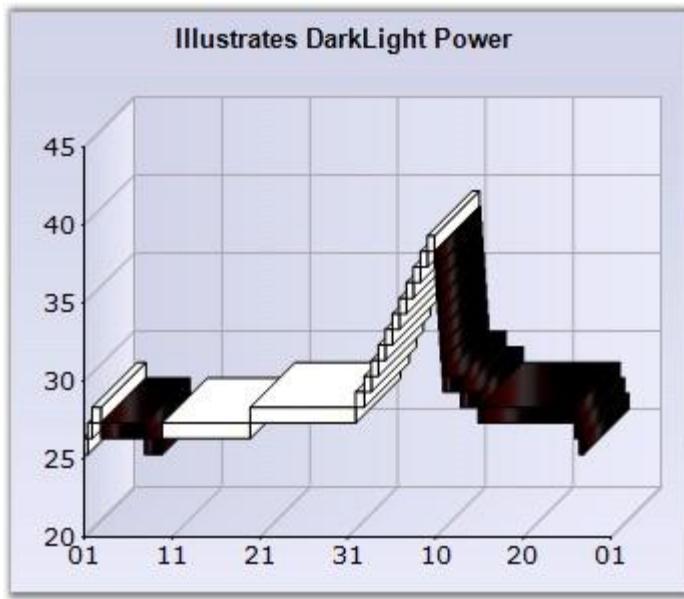


Figure 110: Renko Chart with DarkLightPower set to 200

**See Also**

[Renko Chart](#)

#### 4.5.1.10 DisplayShadow

Specifies if shadow should be displayed for the series.

Details	
<b>Possible Values</b>	True, False
<b>Default Value</b>	False
<b>2D / 3D Limitations</b>	Only 2D

<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	Area Chart, Bar Chart, Bubble Chart, Column Chart, Stacking Column Chart, Stacking Column100 Chart, Line Chart, Spline Chart, Rotated Spline chart, Stepline Chart, Candle Chart, Kagi Chart, Point and Figure Chart, Renko Chart, Threeline Break Charts, Gantt Chart, Histogram chart, Tornado Chart, Combination Chart, Box and Whisker Chart, Pie Chart, Polar And Radar Chart, Step Area Chart

Here is some sample code.

### **Series Wide Setting**

#### **[C#]**

```
this.chartControl1.Series[0].Style.DisplayShadow = true;

//To display shadow for specific data points use Styles[]
series1.Styles[0].DisplayShadow = true;
```

#### **[VB.NET]**

```
Me.chartControl1.Series(0).Style.DisplayShadow = True

'To display shadow for specific data points use Styles[]
series1.Styles(0).DisplayShadow = True
```

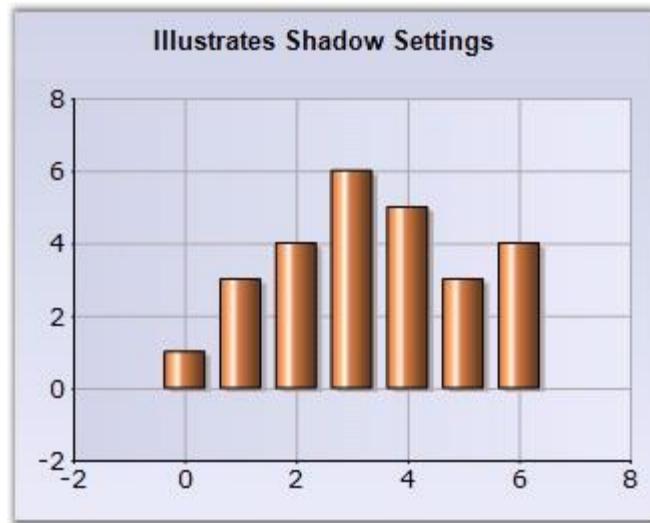


Figure 111: Line Chart with Shadow Series

### Specific Data Point Setting

[C#]

```
this.chartControl1.Series[0].Styles[0].DisplayShadow = true;  
this.chartControl1.Series[0].Styles[1].DisplayShadow = true;
```

[VB .NET]

```
Me.chartControl1.Series(0).Styles(0).DisplayShadow = True  
Me.chartControl1.Series(0).Styles(1).DisplayShadow = True
```

### See Also

[Line Charts](#), [Area Chart](#), [Bubble Chart](#), [Column Chart](#), [Stacking Column Chart](#), [Stacking Column100 Chart](#), [Bar Chart](#), [Pie Chart](#), [Candle Chart](#), [Kagi Chart](#), [Point and Figure Chart](#), [Renko Chart](#), [Three line Break Chart](#), [Gantt Chart](#), [Histogram chart](#), [Tornado Chart](#), [Combination Chart](#), [Box and Whisker Chart](#), [Polar And Radar Chart](#), [Step Area Chart](#)

### 4.5.1.11 DisplayText

Indicates whether a label indicating the data point value should be displayed at the data points.

Details

<b>Possible Values</b>	True, False
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	All Chart types

Here is some sample code.

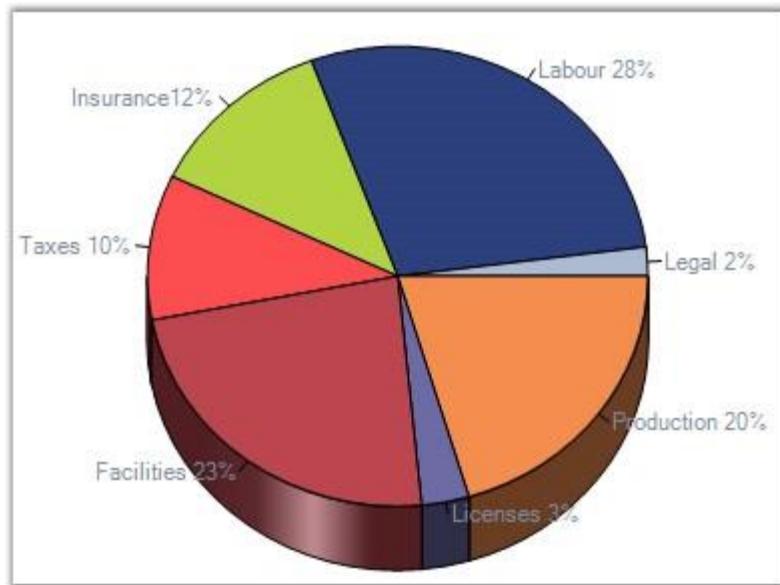
#### **Series wide setting**

**[C#]**

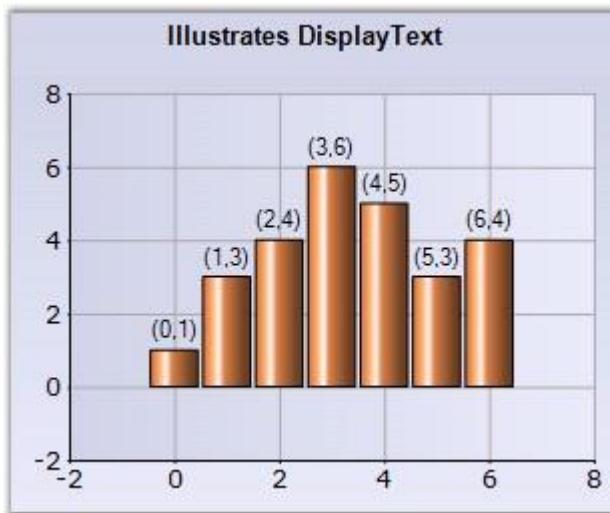
```
// Enabling DisplayText  
this.chartControl1.Series[0].Style.DisplayText = true;  
this.chartControl1.Series[0].Style.TextColor = Color.LightSlateGray;
```

**[VB .NET]**

```
' Enabling DisplayText  
Me.chartControl1.Series(0).Style.DisplayText = True  
Me.chartControl1.Series(0).Style.TextColor = Color.LightSlateGray
```



*Figure 112: DisplayText in Pie Chart*



*Figure 113: DisplayText in Column Chart*

### Specific Data Point Setting

To specify text for specific points, use the below code.

```
[C#]  
// Enabling DisplayText for the first data point
```

```
this.chartControl1.Series[0].Styles[0].DisplayText = true;
this.chartControl1.Series[0].Styles[0].TextColor =
Color.LightSlateGray;
```

**[VB.NET]**

```
'Enabling DisplayText for the first data point
Me.chartControl1.Series(0).Styles(0).DisplayText = True
Me.chartControl1.Series(0).Styles(0).TextColor = Color.LightSlateGray
```

## See Also

[Chart Types](#)

### 4.5.1.12 DoughnutCoefficient

Specifies the percentage of the overall radius of the chart that will be used for the Doughnut center hole. For example, if it is set to 0, the doughnut hole will not exist, therefore, the chart will look like a Pie chart.

Details	
<b>Possible Values</b>	Ranges from 0.0 to 0.9
<b>Default Value</b>	0
<b>2D / 3D Limitations</b>	No.
<b>Applies to Chart Element</b>	All series.
<b>Applies to Chart Types</b>	Doughnut Chart, Pie Chart.

PieCharts with a **DoughnutCoefficient** specified will be rendered as doughnuts. By default, this value is set to 0.0 and hence the chart will be rendered as a full pie.

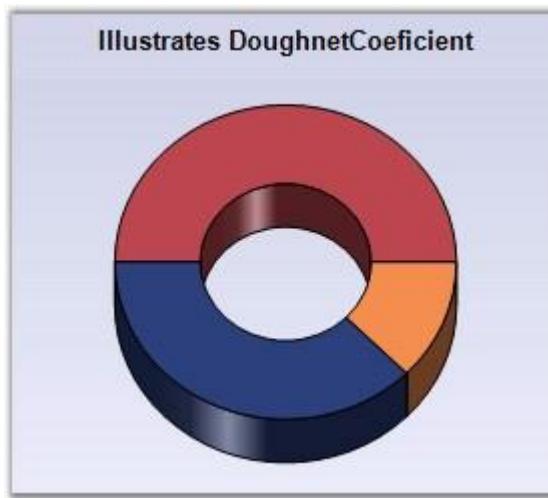
The DoughnutCoefficient property specifies the fraction of the radius occupied by the doughnut whole. Hence the value can range from 0.0 to 0.9.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.PieItem.DoughnutCoefficient =  
0.5f;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.PieItem.DoughnutCoefficient =  
0.5f
```



*Figure 114: Pie Chart with DoughnutCoefficient Property Set*

**See Also**

[Doughnut Chart](#), [Pie Chart](#)

#### 4.5.1.13 DrawColumnSeparatingLines

The drawing of separating line between columns is controlled by this property.

Details	
Possible Values	True or False
Default Value	False

<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Column Chart and Bar Chart

Here is some sample code.

**[C#]**

```
this.chartControl1.Series[0].DrawColumnSeparatingLines = true;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).DrawColumnSeparatingLines = True
```

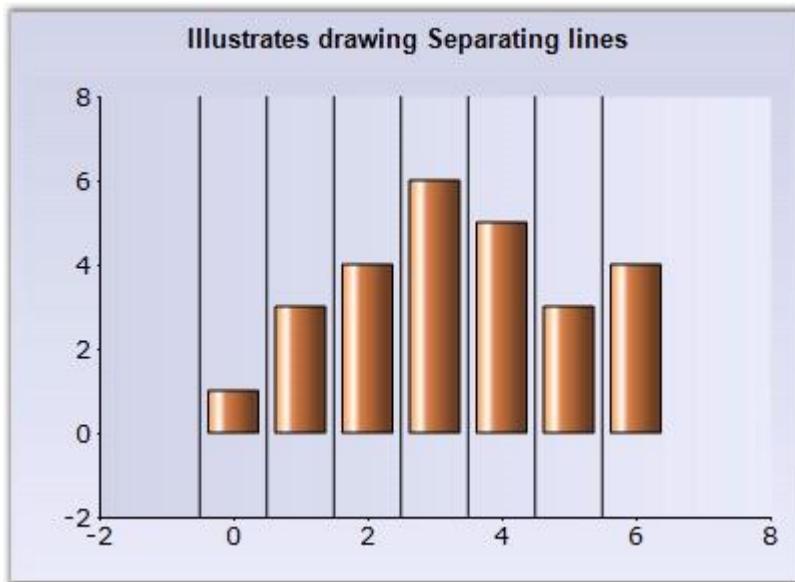


Figure 115: Column Chart without ColumnSeparatingLines

**See Also**

[Column Chart](#), [Bar Chart](#)

#### 4.5.1.14 DrawErrorBars

Error Bars are used to indicate a degree of uncertainty in the plotted data through a bar indicating an "error range".

The 2nd y value is used to indicate the error range. For example, a value of 5 indicates an error range of -5 to +5 from the specified y value.

Details	
<b>Possible Values</b>	True or False
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Column Chart, Line Chart and HiLo Chart

Here is some sample code.

**[C#]**

```
// Generating Series
ChartSeries series = this.chartControl1.Model.NewSeries("Sales",
ChartSeriesType.Column);
// 2nd Y value indicates the error range
series.Points.Add(1, new double[] { 20, 5 });
series.Points.Add(2, new double[] { 70, 6 });
series.Points.Add(3, new double[] { 10, 3 });
series.Points.Add(4, new double[] { 40, 6 });
series.Text = series.Name;
// Adding Series to the Chart
this.chartControl1.Series.Add(series);
// Specifies the Error Bar in Column chart.
this.chartControl1.Series[0].DrawErrorBars = true;
```

**[VB .NET]**

```
' Generating Series
series As ChartSeries = Me.chartControl1.Model.NewSeries("Sales",
ChartSeriesType.Column)
' 2nd Y value indicates the error range
```

```
series.Points.Add(1, New Double() { 20, 5 })
series.Points.Add(2, New Double() { 70, 6 })
series.Points.Add(3, New Double() { 10, 3 })
series.Points.Add(4, New Double() { 40, 6 })
series.Text = series.Name
' Adding Series to the Chart
Me.chartControl1.Series.Add(series)
' Specifies the Error Bar in Column chart.
Private Me.chartControl1.Series(0).DrawErrorBars = True
```

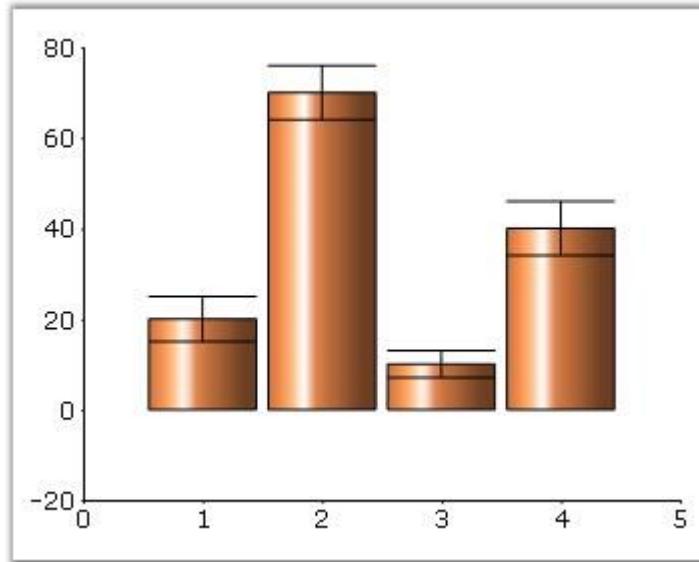


Figure 116: ColumnChart with ErrorBars



*Figure 117: Line Chart with ErrorBars and ErrorBarsSymbolShape="Diamond"*

### ErrorBar Orientation

Orientation of the ErrorBars can be specified in the **ErrorBars.Orientation** property. It can be *Vertical* or *Horizontal*.

#### [C#]

```
//Creates a New Series
ChartSeries s1 = new ChartSeries("series");
//Points for the Series
s1.Points.Add(10, new double[] {20, 2, 2});
s1.Points.Add(20, new double[] {70, 2, 2});
s1.Points.Add(30, new double[] {10, 2, 2});
s1.Points.Add(40, new double[] {40, 2, 2});
s1.Points.Add(40, new double[] {40, 2, 2});
s1.Text = s1.Name;

//Type of Series
s1.Type = ChartSeriesType.Line;
s1.ConfigItems.ErrorBars.Enabled = true;

// Set the orientation to horizontal
s1.ConfigItems.ErrorBars.Orientation = ChartOrientation.Horizontal;
s1.ConfigItems.ErrorBars.SymbolShape = ChartSymbolShape.None;

s1.Style.Interior = new Syncfusion.Drawing.BrushInfo(Color.Red);
this.chartControl1.PrimaryXAxis.DrawGrid = false;
this.chartControl1.PrimaryYAxis.DrawGrid = false;

this.chartControl1.Series.Add(s1);
```

#### [VB .NET]

```
'Creates a New Series
Dim s1 As New ChartSeries("series")
'Points for the Series
s1.Points.Add(10, New Double() {20, 2, 2})
s1.Points.Add(20, New Double() {70, 2, 2})
s1.Points.Add(30, New Double() {10, 2, 2})
s1.Points.Add(40, New Double() {40, 2, 2})
s1.Points.Add(40, New Double() {40, 2, 2})
s1.Text = s1.Name
```

```
'Type of Series
s1.Type = ChartSeriesType.Line
s1.ConfigItems.ErrorBars.Enabled = True

' Set the orientation to horizontal
s1.ConfigItems.ErrorBars.Orientation = ChartOrientation.Horizontal
s1.ConfigItems.ErrorBars.SymbolShape = ChartSymbolShape.None

s1.Style.Interior = New Syncfusion.Drawing.BrushInfo(Color.Red)
Me.chartControll1.PrimaryXAxis.DrawGrid = False
Me.chartControll1.PrimaryYAxis.DrawGrid = False

Me.chartControll1.Series.Add(s1)
```

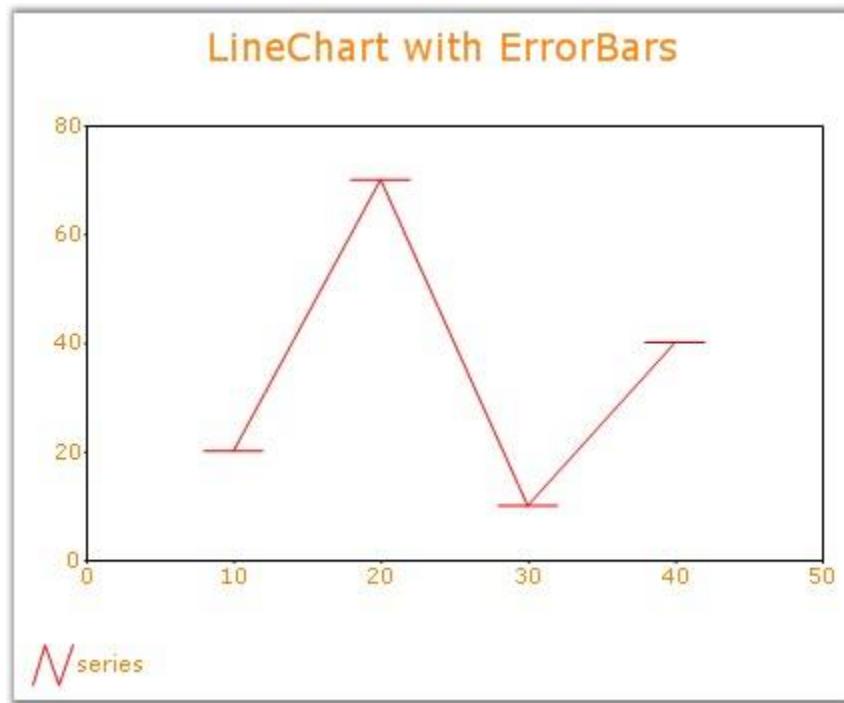


Figure 118: Errorbar Orientation = "Horizontal"

## See Also

[Line Chart](#), [Column Chart](#), [Hi Lo Chart](#), [ErrorBarsSymbolShape](#)

#### 4.5.1.15 DrawHistogramNormalDistribution

The normal distribution curve is drawn by setting this property of the ChartSeries class to **true**.

Details	
<b>Possible Values</b>	True or False
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Histogram Chart

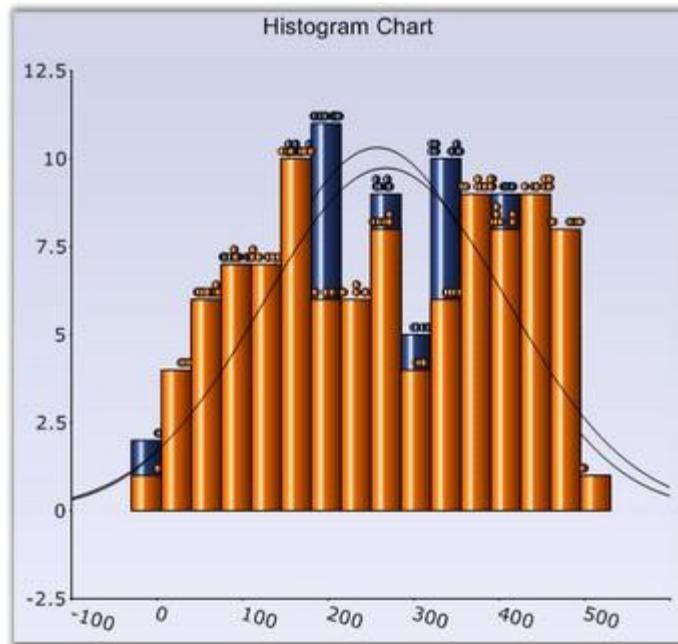
Here is some sample code.

**[C#]**

```
// This draws the normal distribution curve for the histogram chart.  
series2.DrawHistogramNormalDistribution = true;  
  
// Set the desired number of intervals required for the histogram  
// chart.  
series2.NumberOfHistogramIntervals = 10;
```

**[VB.NET]**

```
' This draws the normal distribution curve for the histogram chart.  
series2.DrawHistogramNormalDistribution = True  
  
' Set the desired number of intervals required for the histogram chart.  
series2.NumberOfHistogramIntervals = 10
```



*Figure 119: Histogram Chart with Normal Distribution Curve*

#### **See Also**

[Histogram Chart](#)

#### **4.5.1.16 DrawSeriesNameInDepth**

Indicates whether to draw series name at opposed position to origin, along x-axis.

Details	
<b>Possible Values</b>	True or False
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	3D Only

<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	All chart types

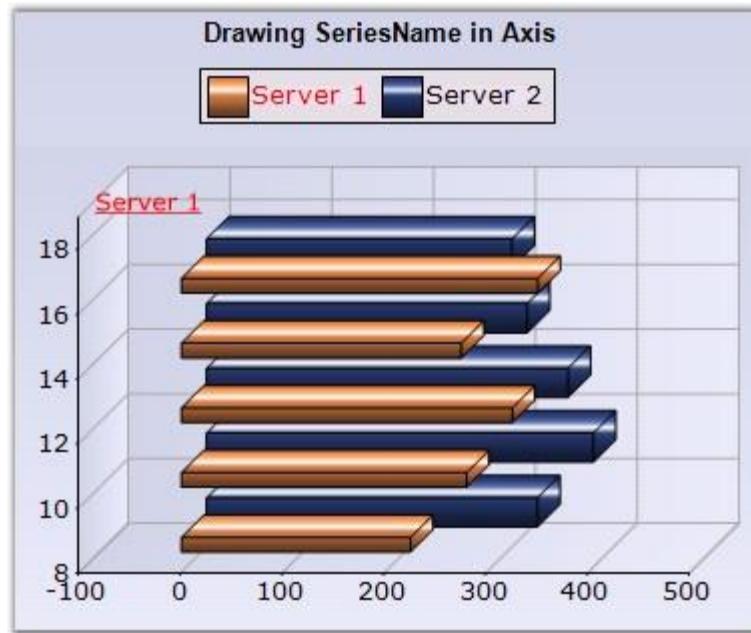
Here is some sample code.

**[C#]**

```
// Specified 3D View
this.chartControl1.Series3D = true;
// Setting Text Format
this.chartControl1.Series[0].Style.Font.FontStyle =
FontStyle.Underline;
this.chartControl1.Series[0].Style.TextColor = Color.Black;
this.chartControl1.Series[0].Style.Font.Size = 7;
this.chartControl1.Series[0].Style.Font.Facename = "Times New Roman";
// Set SeriesNameDepth as True
this.chartControl1.Series[0].DrawSeriesNameInDepth = true;
```

**[VB .NET]**

```
' Specified 3D View
Me.chartControl1.Series3D = True
' Setting Text Format
Me.chartControl1.Series(0).Style.Font.FontStyle = FontStyle.Underline
Me.chartControl1.Series(0).Style.TextColor = Color.Black
Me.chartControl1.Series(0).Style.Font.Size = 7
Me.chartControl1.Series(0).Style.Font.Facename = "Times New Roman"
' Set SeriesNameDepth as True
Me.chartControl1.Series(0).DrawSeriesNameInDepth = True
```



*Figure 120: DrawSeriesNameInDepth in BarChart*

#### See Also

[Chart Types](#)

#### 4.5.1.17 DropSeriesPoints

The chart also provides you an option to drop some points (by not drawing some points) while rendering large number of chart points.

You can enable this, by setting the **DropSeriesPoints** property to **true**.

[C#]

```
this.chartControl1.DropSeriesPoints = true;
```

[VB .NET]

```
Me.chartControl1.DropSeriesPoints = True
```

#### 4.5.1.18 ElementBorders

Gets / sets the border settings for elements associated with the chart point. You can specify the inner and outer border. It is currently used only by symbols rendered by the ChartPoint (inherited from ChartStyleInfo).

Details	
<b>Possible Values</b>	Border setting object
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>Weight</b> – Thin</li> <li>• <b>Width value</b> – 1</li> <li>• <b>Style</b> - Standard</li> </ul>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	Area Charts, Bar Charts, Bubble Chart, Column Charts, Line Charts, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Tornado Chart, Polar and Radar Chart

Here is some sample code.

#### Series Wide Setting

[C#]

```
// Setting Symbol for the ChartSeries
this.chartControl1.Series[0].Style.Symbol.Color = Color.Yellow;
this.chartControl1.Series[0].Style.Symbol.Shape =
ChartSymbolShape.InvertedTriangle;
// Setting ElementBorder for a symbol
ChartBordersInfo cbi = new ChartBordersInfo();
cbi.Outer = new ChartBorder(ChartBorderStyle.Solid, Color.White);
cbi.Inner = new ChartBorder(ChartBorderStyle.DashDot, Color.Cyan);
this.chartControl1.Series[0].Style.ElementBorders = cbi;
```

[VB .NET]

```
' Setting Symbol for the ChartSeries
```

```
Me.chartControl1.Series(0).Style.Symbol.Color = Color.Yellow  
Me.chartControl1.Series(0).Style.Symbol.Shape =  
ChartSymbolShape.InvertedTriangle  
' Setting ElementBorder for a symbol  
cbi As ChartBordersInfo = New ChartBordersInfo()  
cbi.Outer = New ChartBorder(ChartBorderStyle.Solid, Color.White)  
cbi.Inner = New ChartBorder(ChartBorderStyle.DashDot, Color.Cyan)  
Me.chartControl1.Series(0).Style.ElementBorders = cbi
```

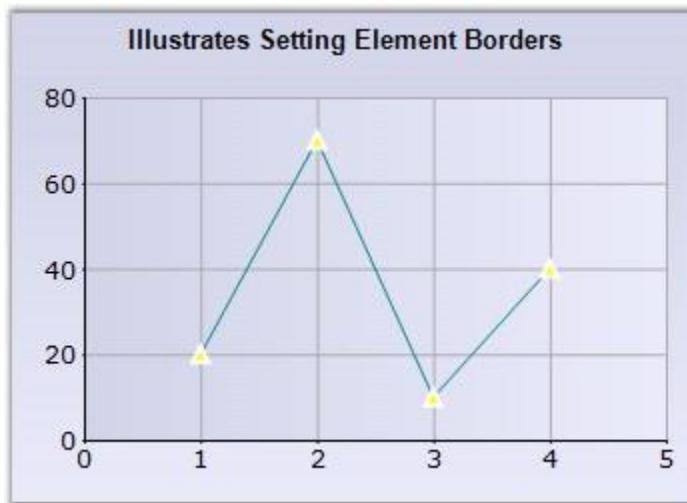


Figure 121: Column Chart with ElementBorder

### Specific Data Point Setting

[C#]

```
//Specifying element border for the first data point Styles(0), second  
//data point Styles(1) and so on..  
this.chartControl1.Series[0].Styles[0].ElementBorders = cbi;
```

[VB .NET]

```
'Specifying element border for the first data point Styles(0), second  
//data point Styles(1) and so on..  
this.chartControl1.Series(0).Styles(0).ElementBorders = cbi
```

### See Also

[Area Charts](#), [Bar Charts](#), [Bubble Chart](#), [Column Charts](#), [Line Charts](#), [Candle Chart](#), [Renko chart](#), [Three Line Break Chart](#),  
[Box and Whisker Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Polar and Radar Chart](#)

#### 4.5.1.19 EnablePhongStyle

Specifies if the PhongStyle is enabled.

Details	
<b>Possible Values</b>	True or False
<b>Default Value</b>	<b>True</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Bubble Chart

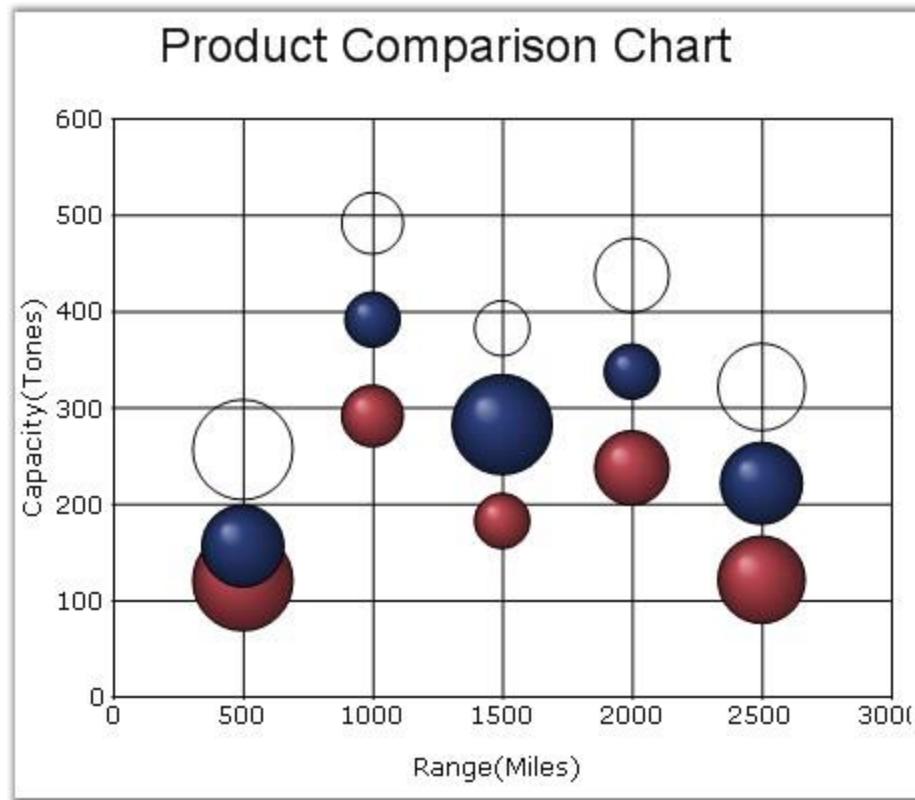
Here is some sample code.

[C#]

```
this.chartControl1.Series[0].ConfigItems.BubbleItem.EnablePhongStyle =  
false;
```

[VB .NET]

```
Me.chartControl1.Series(0).ConfigItems.BubbleItem.EnablePhongStyle =  
False
```



*Figure 122: Bubble Chart with PhongStyle disabled for One Series*

#### See Also

[Bubble Chart](#)

#### 4.5.1.20 EnableAreaToolTip

To display proper tooltip for the Area charts, use the **Series.EnableAreaToolTip** property. When this property is enabled, the series index and the point index will be returned, which will be suitable for the various PointsToolTipFormats also.

This splits up the region between two points into two parts while hovering the mouse on the region and displays the tooltip with respect to the nearby chart point.

[C#]

```
this.chartControl1.Series[0].EnableAreaToolTip = true;
```

[VB.NET]

```
Me.chartControl1.Series(0).EnableAreaToolTip = True
```

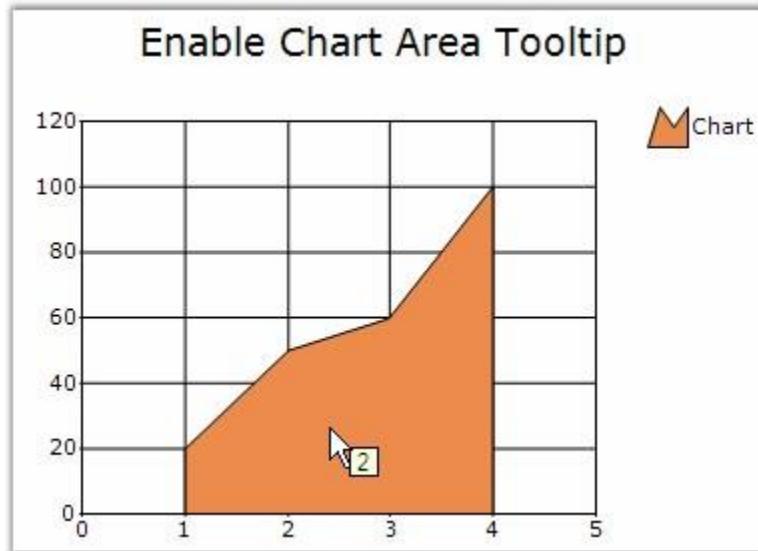


Figure 123: *EnableAreaToolTip = "True"*

#### 4.5.1.21 ErrorBarsSymbolShape

This property determines the shape of the error bar symbol when DrawErrorBars is **true**.

Details	
Possible Values	
	<ul style="list-style-type: none"><li>• None - No marker will be shown.</li><li>• Line - A Line will be drawn as the marker.</li><li>• Square - A Square will be drawn as the marker.</li><li>• Circle - A Circle will be drawn as the marker.</li><li>• Diamond - A Diamond will be drawn as the marker.</li><li>• Triangle - A Triangle will be drawn as the marker.</li></ul>

	<ul style="list-style-type: none"> <li>VertLine - A VerticalLine will be drawn as the marker.</li> <li>Cross - A Cross will be drawn as the marker.</li> <li>Hexagon - An Hexagon will be drawn as the marker.</li> <li>HorizLine - An Horizontal Line will be drawn as the marker.</li> <li>Image - An Image will be drawn as the marker.</li> <li>InvertedTriangle - A InvertedTriangle will be drawn as the marker.</li> <li>Pentagon - A Pentagon will be drawn as the marker.</li> <li>Star - A Star will be drawn as the marker.</li> </ul>
<b>Default Value</b>	<b>Diamond</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Line Chart

Here is some sample code.

**[C#]**

```
this.chartControl1.Series[0].DrawErrorBars = true;
this.chartControl1.Series[0].ErrorBarsSymbolShape =
ChartSymbolShape.Circle;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).DrawErrorBars = true
Me.chartControl1.Series(0).ErrorBarsSymbolShape =
ChartSymbolShape.Circle
```



Figure 124: Line Chart with ErrorBarSymbol set to "Circle"

#### See Also

[Line Chart](#), [DrawErrorBars](#)

#### 4.5.1.22 ExplodedAll

Indicates whether to explode all slice in the Pie or Doughnut chart.

Details	
Possible Values	True, False
Default Value	<b>False</b>
2D / 3D Limitations	No
Applies to Chart Element	All series
Applies to Chart Types	Pie Chart, Doughnut Chart

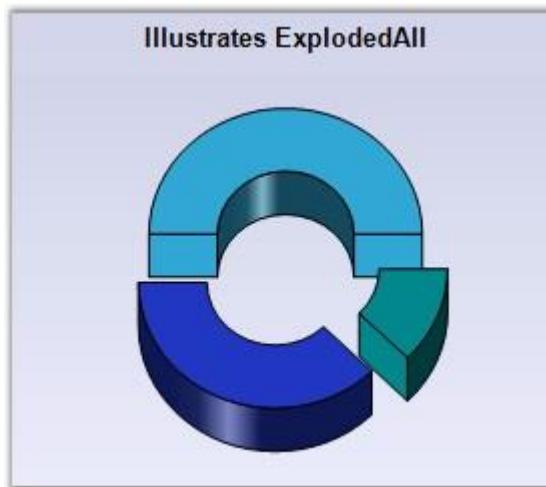
Here is some sample code.

[C#]

```
this.chartControl1.Series[0].ExplodedAll = true;
```

[VB .NET]

```
Me.chartControl1.Series(0).ExplodedAll = True
```



*Figure 125: Exploded Pie Chart*

#### See Also

[Doughnut Chart, Pie Chart](#)

#### 4.5.1.23 ExplodedIndex

Gets / sets the Index point that is to be used when a point is to be exploded from the main display.

Details	
<b>Possible Values</b>	An integer indicating the index of the slice to be exploded.
<b>Default Value</b>	-1
<b>2D / 3D Limitations</b>	No

<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Pie Chart, Doughnut Chart

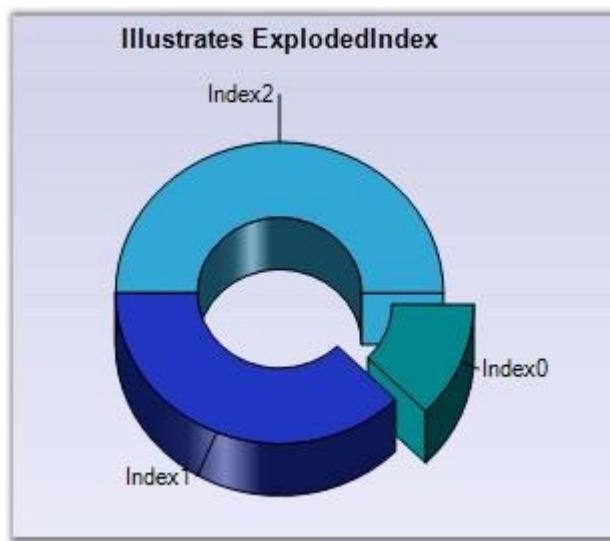
Here is some sample code.

**[C#]**

```
this.chartControl1.Series[0].ExplodedIndex = 0;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ExplodedIndex = 0
```



*Figure 126: Pie Chart with Exploded Index*

#### **See Also**

[Doughnut Chart, Pie Chart](#)

#### **4.5.1.24 ExplosionOffset**

Gets / sets the offset value that is to be used when slices are to be exploded in a pie chart.

Details	
Possible Values	Float type values
Default Value	20
2D / 3D Limitations	No
Applies to Chart Element	All series
Applies to Chart Types	Pie Chart, Doughnut Chart

Here is some sample code.

[C#]

```
this.chartControl1.Series[0].ExplodedAll = true;
this.chartControl1.Series[0].ExplosionOffset = 30f;
```

[VB .NET]

```
Me.chartControl1.Series[0].ExplodedAll = True
Me.chartControl1.Series(0).ExplosionOffset = 30f
```

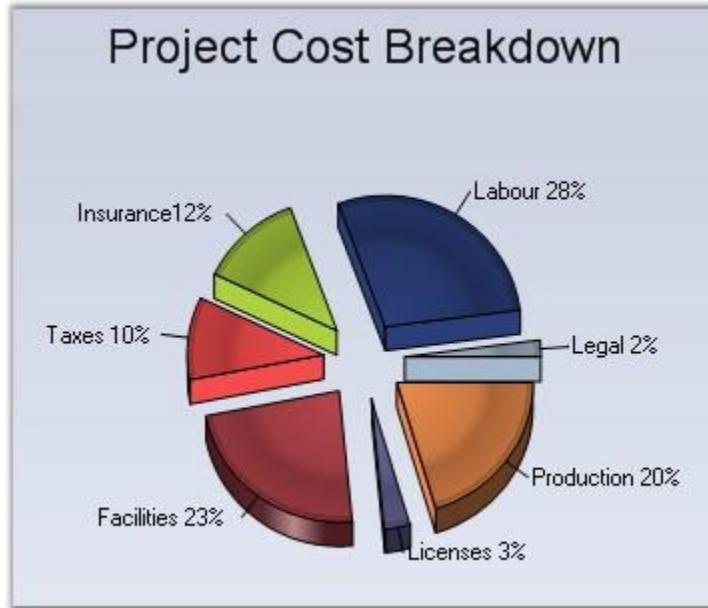


Figure 127: Exploded Pie Chart

#### See Also

[Doughnut Chart](#), [Pie Chart](#)

### 4.5.1.25 FancyToolTip

Defines the styles for a fancy tooltip. These styles include font, marker style, symbol shape, back color and other related styles.

Details	
<b>Possible Values</b>	Specifies symbol, symbol styles for the ToolTip.
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>Visible</b> - False</li> <li>• <b>Angle</b> - 15</li> <li>• <b>Alignment</b> - Left</li> <li>• <b>ForeColor</b> - Color.Black</li> <li>• <b>BackColor</b> - Color.Info</li> <li>• <b>SymbolColor</b> - Color.Red</li> <li>• <b>Font</b> - Arial, 8 pt</li> <li>• <b>Symbol Size</b> - (10,10)</li> <li>• <b>Symbol</b> - Circle</li> <li>• <b>Style</b> - SmoothRectangle</li> </ul>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	All Chart Types

Here is some sample code.

[C#]

```
this.chartControl1.Series[0].FancyToolTip.Angle = 180;
this.chartControl1.Series[0].FancyToolTip.Style =
MarkerStyle.SmoothRectangle;
this.chartControl1.Series[0].FancyToolTip.Symbol =
ChartSymbolShape.Hexagon;
this.chartControl1.Series[0].FancyToolTip.Visible = true;
```

**[VB.NET]**

```
Me.chartControl1.Series(0).FancyToolTip.Angle = 180  
Me.chartControl1.Series(0).FancyToolTip.Style =  
MarkerStyle.SmoothRectangle  
Me.chartControl1.Series(0).FancyToolTip.Symbol =  
ChartSymbolShape.Hexagon  
Me.chartControl1.Series(0).FancyToolTip.Visible = True
```

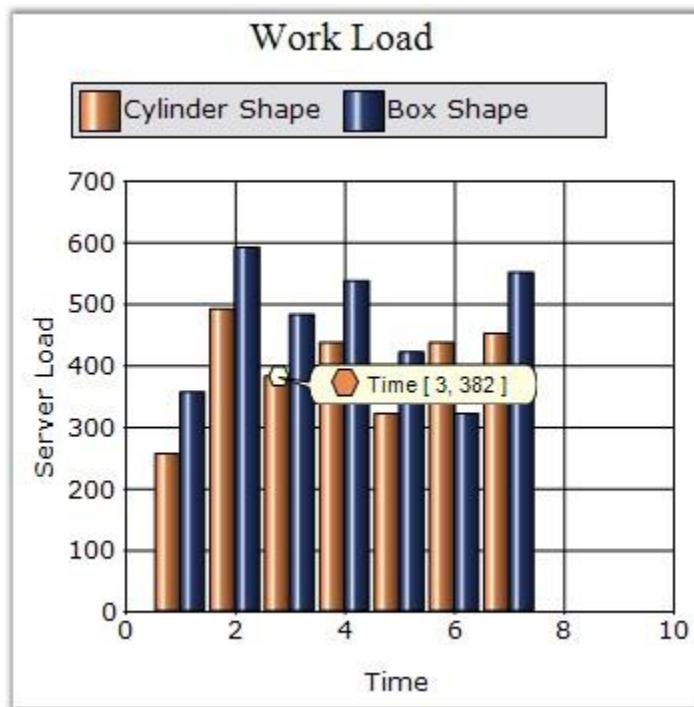


Figure 128: StackingBar Chart with FancyToolTip

**See Also**

[Chart Types](#)

#### 4.5.1.26 FigureBase

Specifies the drawing style for the funnel or pyramid chart base.

**Details**

<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>Circle</b> - Renders the chart with a circular base.</li> <li>• <b>Square</b> - Renders the chart with a square base.</li> </ul>
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>Funnel Chart</b> - Circle</li> <li>• <b>Pyramid Chart</b> - Square</li> </ul>
<b>2D / 3D Limitations</b>	3D Only
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Funnel and Pyramid

Here is some sample code.

#### [C#]

```
// Setting FigureBase For Pyramid Chart
this.chartControl1.Series[0].ConfigItems.PyramidItem.FigureBase =
ChartFigureBase.Circle;
this.chartControl1.Series[0].ConfigItems.PyramidItem.FigureBase =
ChartFigureBase.Square;

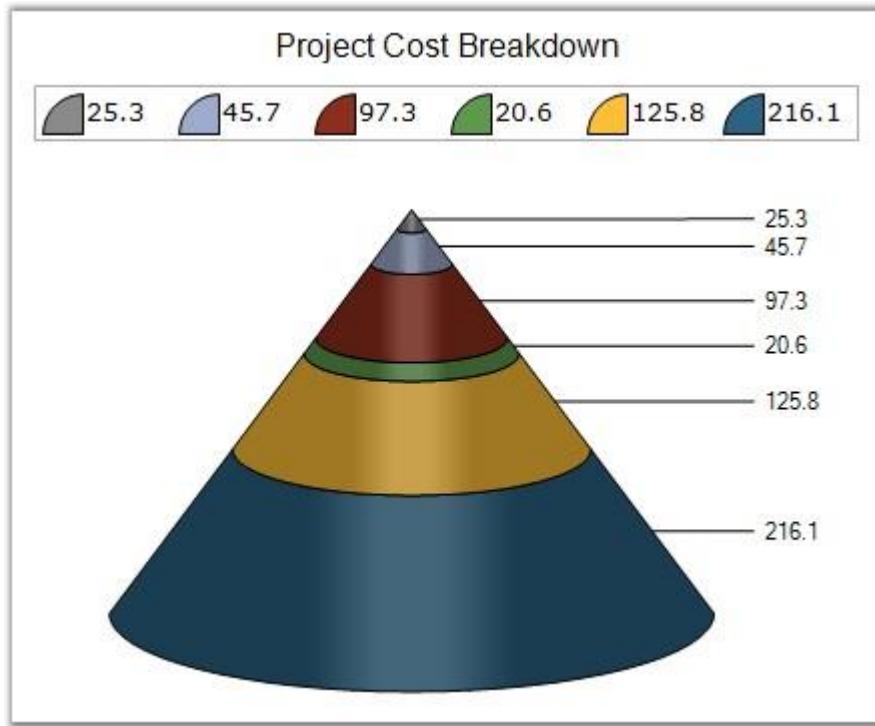
// Setting FigureBase For Funnel Chart
this.chartControl1.Series[0].ConfigItems.FunnelItem.FigureBase =
ChartFigureBase.Circle;
this.chartControl1.Series[0].ConfigItems.FunnelItem.FigureBase =
ChartFigureBase.Square;
```

#### [VB.NET]

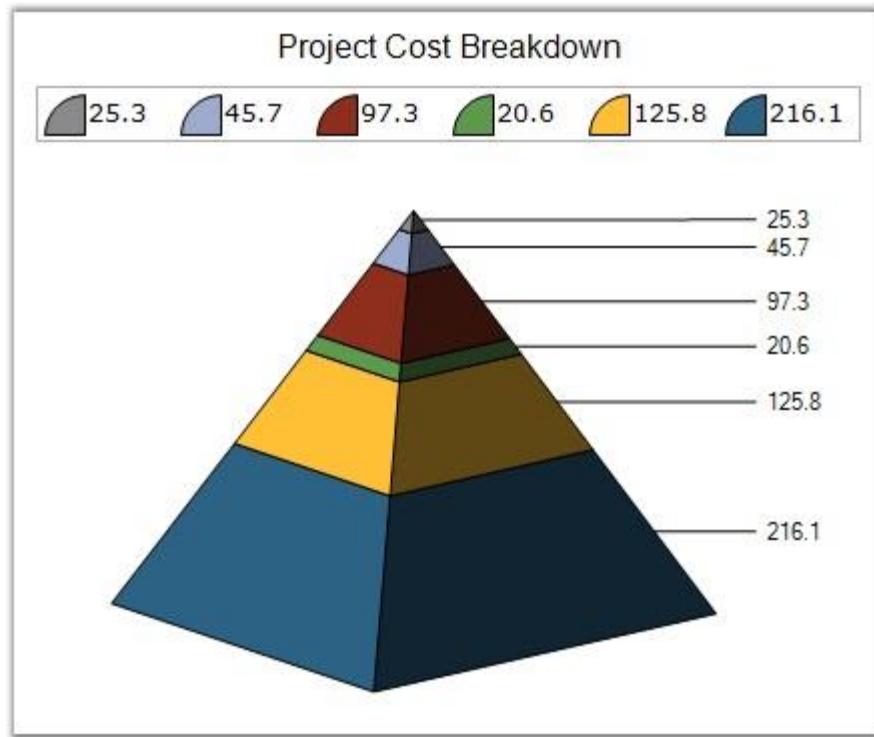
```
' Setting FigureBase For Pyramid
Me.chartControl1.Series(0).ConfigItems.PyramidItem.FigureBase=ChartFigureBase.Circle
Me.chartControl1.Series(0).ConfigItems.PyramidItem.FigureBase=ChartFigureBase.Square

' Setting FigureBase For Funnel Chart
Me.chartControl1.Series(0).ConfigItems.FunnelItem.FigureBase =
ChartFigureBase.Circle
Me.chartControl1.Series(0).ConfigItems.FunnelItem.FigureBase =
ChartFigureBase.Square
```

### Pyramid Chart

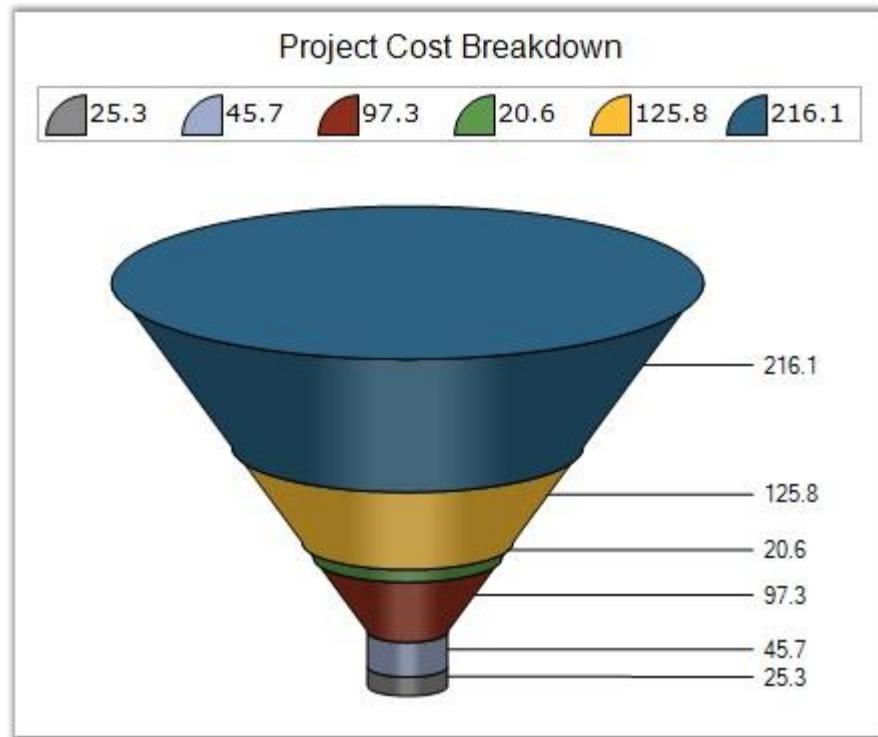


*Figure 129: Pyramid Chart with Figure Base = "Circle"*



*Figure 130: Pyramid Chart with Figure Base ="Square"*

### **Funnel Chart**



*Figure 131: Funnel Chart with Figure Base = "Circle"*

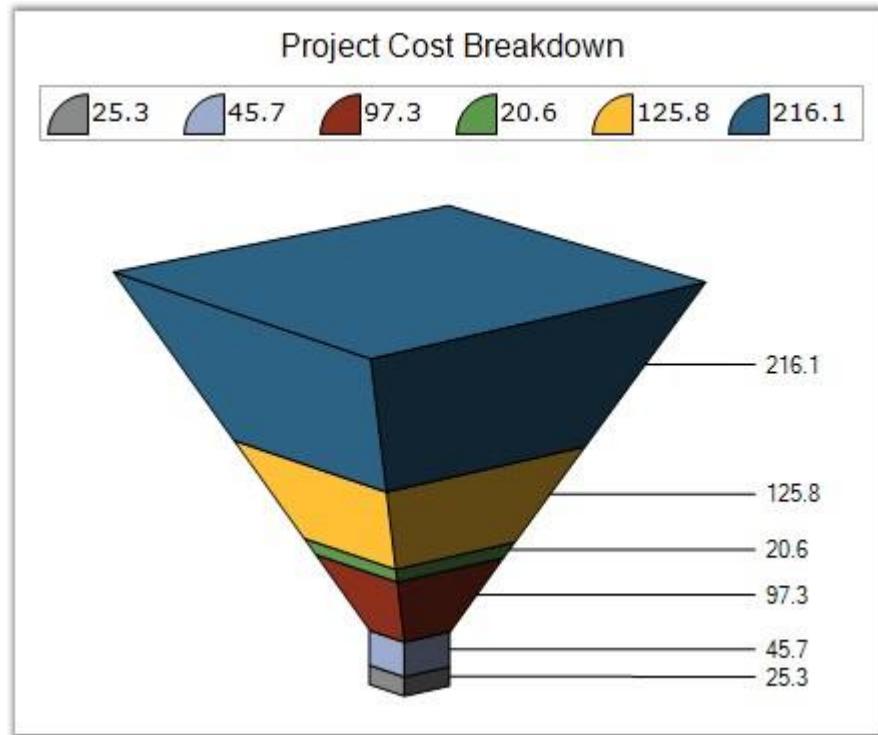


Figure 132: Funnel Chart with Figure Base ="Square"

**See Also**

[Pyramid Chart, Funnel Chart](#)

**4.5.1.27 FillMode**

Specifies how the slice interior should be filled with gradient colors.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>AllPie</b> - Controls the interior shape style of All PieItem.</li> <li>• <b>EveryPie</b> - Controls the interior shape style of Every PieItem.</li> </ul>
<b>Default Value</b>	AllPie
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Pie Chart

Here is some sample code.

**[C#]**

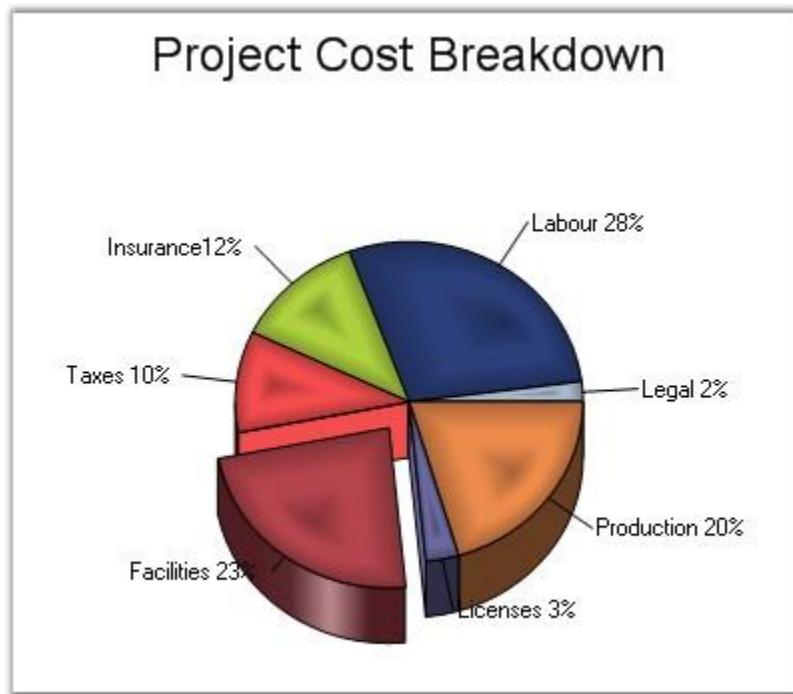
```
// Setting Pietype
this.chartControl1.Series[0].ConfigItems.PieItem.PieType =
ChartPieType.Round;
// Setting the interiors of shapes in this GraphicsPath object are
filled.
this.chartControl1.Series[0].ConfigItems.PieItem.FillMode =
ChartPieFillMode.EveryPie;
```

**[VB .NET]**

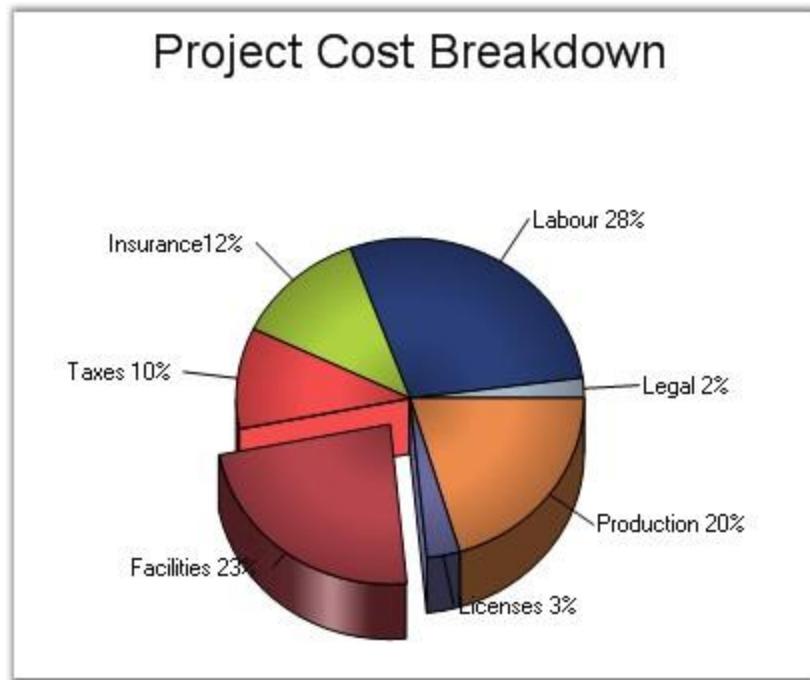
```
' Setting Pietype
Me.chartControl1.Series(0).ConfigItems.PieItem.PieType =
ChartPieType.Round
' Setting the interiors of shapes in this GraphicsPath object are
```

filled.

```
Me.chartControll1.Series(0).ConfigItems.PieItem.FillMode =  
ChartPieFillMode.EveryPie
```



*Figure 133: Pie Chart with "EveryPie" FillMode*



*Figure 134: Pie Chart with "AllPie" FillMode*

#### See Also

[Pie Chart](#)

#### 4.5.1.28 FunnelMode

Gets or sets the chart funnel mode.

Details	
Possible Values	<ul style="list-style-type: none"><li>• <b>YIsWidth</b> - DataPoint y-value controls the radius of the funnel segment.</li><li>• <b>YIsHeight</b> - DataPoint y-value controls the height of the funnel segment.</li></ul>

<b>Default Value</b>	<b>YIsHeight</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Funnel Chart

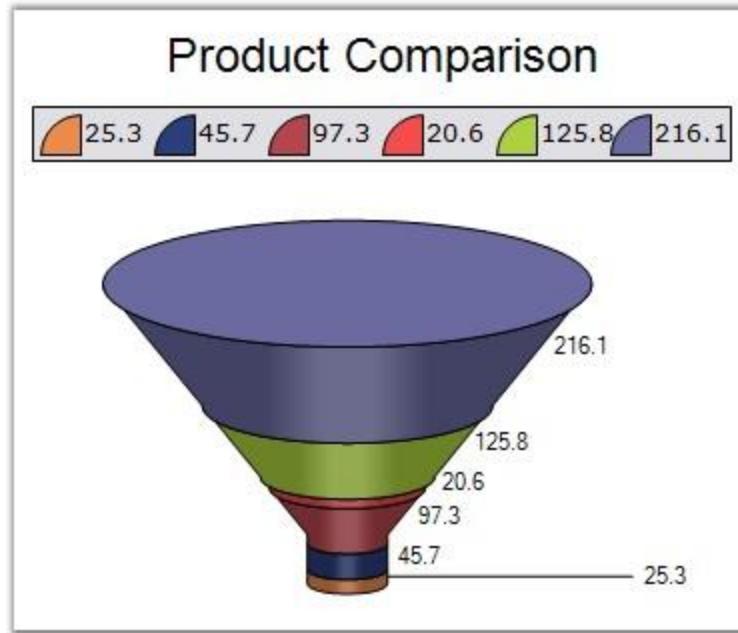
Here is some sample code.

**[C#]**

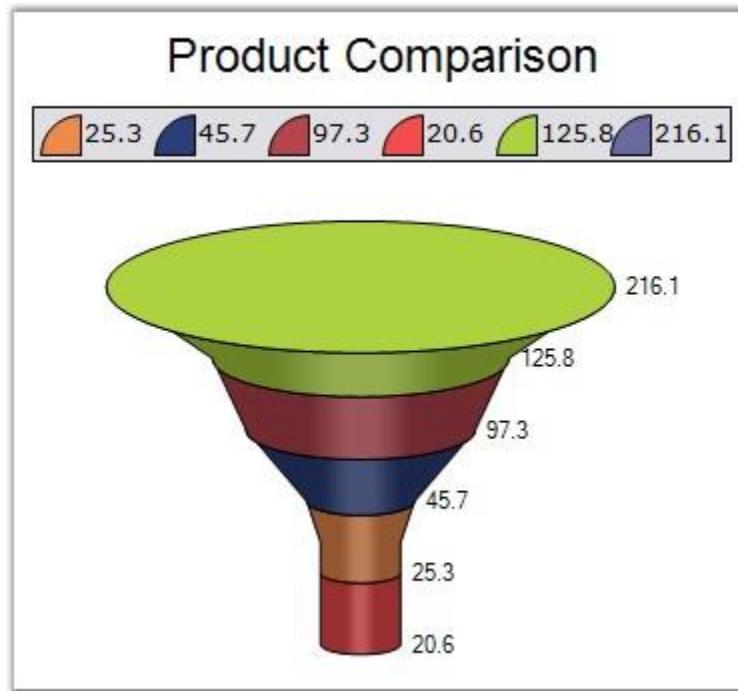
```
this.chartControll1.Series[0].ConfigItems.FunnelItem.FunnelMode =  
ChartFunnelMode.YIsHeight;  
this.chartControll1.Series[0].ConfigItems.FunnelItem.FunnelMode =  
ChartFunnelMode.YIsWidth;
```

**[VB .NET]**

```
Me.chartControll1.Series(0).ConfigItems.FunnelItem.FunnelMode =  
ChartFunnelMode.YIsHeight  
Me.chartControll1.Series(0).ConfigItems.FunnelItem.FunnelMode =  
ChartFunnelMode.YIsWidth
```



*Figure 135: Funnel Chart with FunnelMode as "Height"*



*Figure 136: Funnel Chart with FunnelMode as "Width"*

#### See Also

[Funnel Chart](#)

#### 4.5.1.29 Font

Gets or sets a font object used for drawing the data point labels.

Details	
<b>Possible Values</b>	Specifying font face, size and style.
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>FontStyle</b> - Regular</li> <li>• <b>Face Name</b> - MicroSoft Sans Serif</li> <li>• <b>Size</b> - 8.25</li> </ul>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	All Chart types

Here is some sample code.

**Series Wide Setting****[C#]**

```
this.chartControl1.Series[0].Style.DisplayText = true;
this.chartControl1.Series[0].Style.Font.Bold = true;
this.chartControl1.Series[0].Style.Font.Facename = "Arial";
this.chartControl1.Series[0].Style.Text = "Series 1";
```

**[VB .NET]**

```
Me.chartControl1.Series(0).Style.DisplayText = True  
Me.chartControl1.Series(0).Style.Font.Bold = True  
Me.chartControl1.Series(0).Style.Font.Facename = "Arial"  
Me.chartControl1.Series(0).Style.Text = "Series 1"
```

### Specific Data Point Setting

[C#]

```
//font style set for first data point  
this.chartControl1.Series[0].Styles[0].Font.Bold = true;  
this.chartControl1.Series[0].Styles[0].Font.Facename = "Arial";
```

[VB .NET]

```
'font style set for first data point  
Me.chartControl1.Series(0).Styles(0).Font.Bold = True  
Me.chartControl1.Series(0).Styles(0).Font.Facename = "Arial"
```

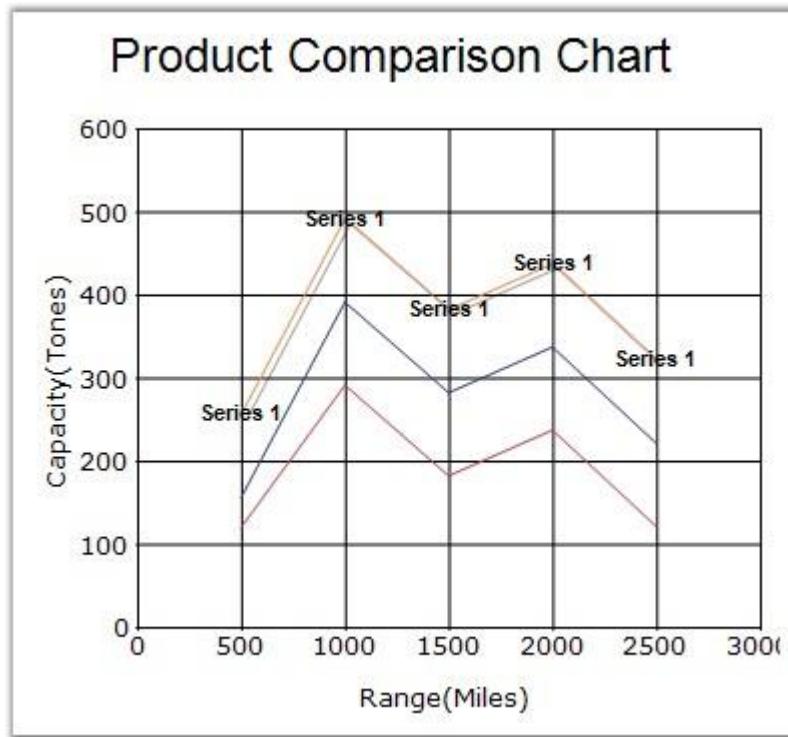


Figure 137: Column Chart with Text

### See Also

[Chart Types](#)

#### 4.5.1.30 GanttDrawMode

Specifies the drawing mode of Gantt chart.

Details	
Possible Values	<ul style="list-style-type: none"> <li>• <b>AutoSizeMode</b> - Plots the Gantt Chart side by side.</li> <li>• <b>CustomPointWidthMode</b> - Plots the Gantt Chart as Overlapped.</li> </ul>
Default Value	<b>CustomPointWidthMode</b>
2D / 3D Limitations	None
Applies to Chart Element	All series
Applies to Chart Types	Gantt Chart

Here is some sample code.

**[C#]**

```
// Specifies GanttDrawMode as CustomPointWidthMode
this.chartControl1.Series[0].GanttDrawMode =
ChartGanttDrawMode.CustomPointWidthMode;
this.chartControl1.Series[0].Style.PointWidth = 0.7f;
this.chartControl1.Series[1].GanttDrawMode =
ChartGanttDrawMode.CustomPointWidthMode;
this.chartControl1.Series[1].Style.PointWidth = 1f;
```

**[VB .NET]**

```
' Specifies GanttDrawMode as CustomPointWidthMode
Me.chartControl1.Series(0).GanttDrawMode =
ChartGanttDrawMode.CustomPointWidthMode
Me.chartControl1.Series(0).Style.PointWidth = 0.7f
Me.chartControl1.Series(1).GanttDrawMode =
ChartGanttDrawMode.CustomPointWidthMode
```

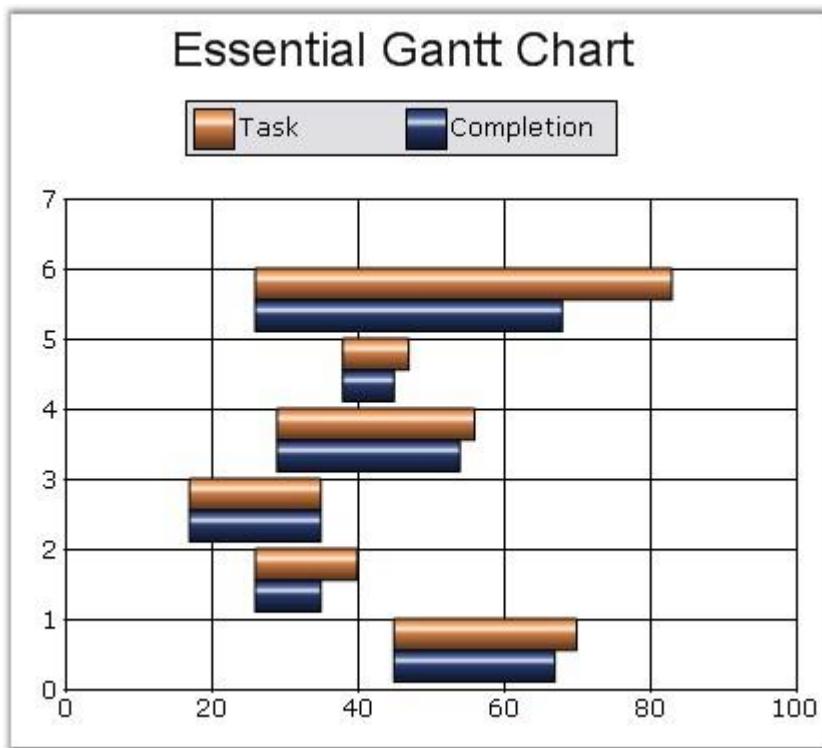
```
Me.chartControl1.Series(1).Style.PointWidth = 1f
```

**[C#]**

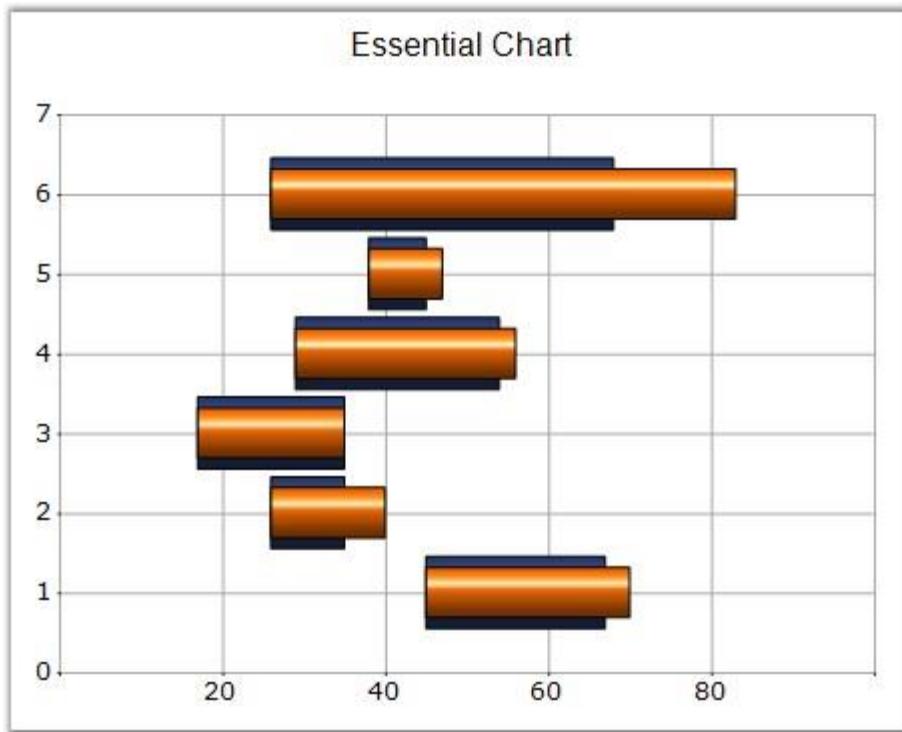
```
// Specifies GanttDrawMode as AutoSizeMode  
this.chartControl1.Series[0].GanttDrawMode =  
ChartGanttDrawMode.AutoSizeMode;  
this.chartControl1.Series[1].GanttDrawMode =  
ChartGanttDrawMode.AutoSizeMode;
```

**[VB .NET]**

```
' Specifies GanttDrawMode as AutoSizeMode  
Me.chartControl1.Series(0).GanttDrawMode =  
ChartGanttDrawMode.AutoSizeMode  
Me.chartControl1.Series(1).GanttDrawMode =  
ChartGanttDrawMode.AutoSizeMode
```



*Figure 138: Gantt Chart with AutoSizeMode*



*Figure 139: Gantt Chart with CustomPointWidthMode*

**See Also**

[Gantt Chart](#)

#### 4.5.1.31 GapRatio

Gets or sets the gap size between funnel chart or pyramid chart segments. Default value is 0.0. The maximum gap size is limited by the number of points.

Details	
Possible Values	Ranges from 0.0
Default Value	0
2D / 3D Limitations	No

<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Funnel Chart, Pyramid Chart

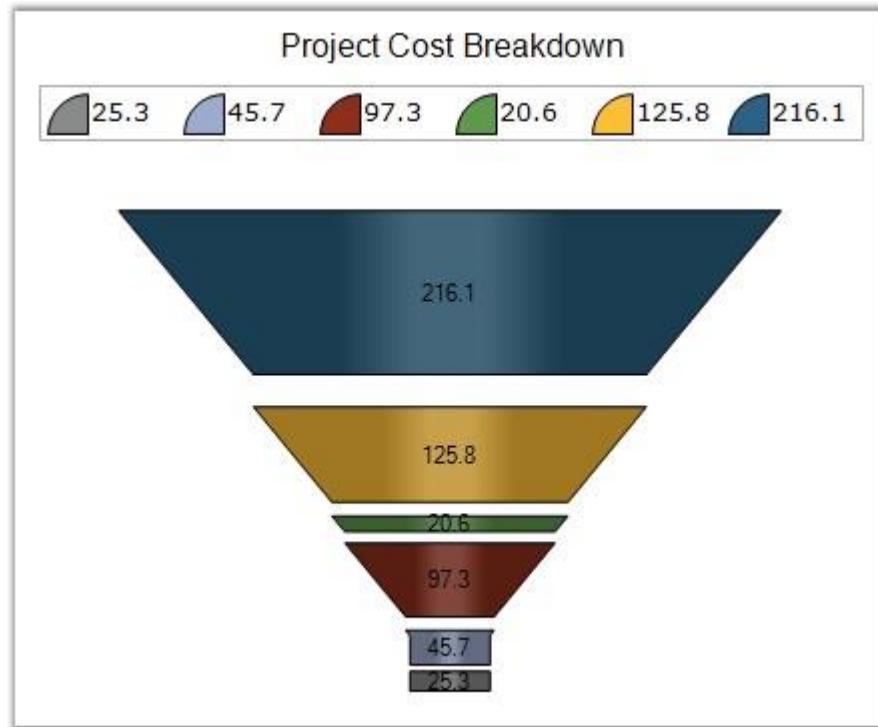
Here is some sample code.

**[C#]**

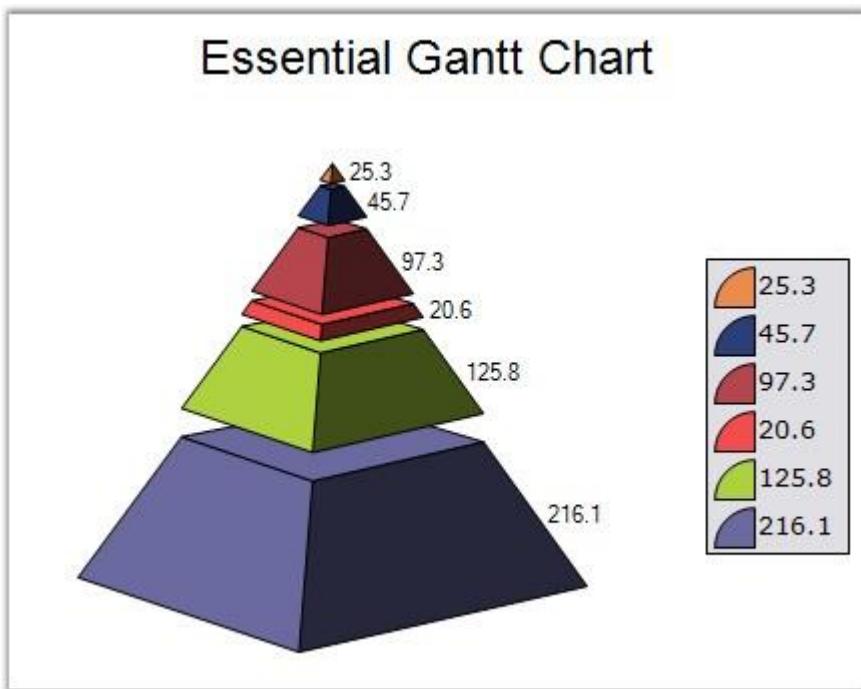
```
// Setting GapRatio for Funnel Chart  
this.chartControl1.Series[0].ConfigItems.FunnelItem.GapRatio = 0.1f;  
  
// Setting GapRatio for Pyramid Chart  
this.chartControl1.Series[0].ConfigItems.PyramidItem.GapRatio = 0.1f;
```

**[VB.NET]**

```
' Setting GapRatio for Funnel Chart  
Me.chartControl1.Series(0).ConfigItems.FunnelItem.GapRatio = 0.1f  
  
' Setting GapRatio for Pyramid Chart  
Me.chartControl1.Series(0).ConfigItems.PyramidItem.GapRatio = 0.1f
```



*Figure 140: Funnel chart with 0.1 GapRatio*



*Figure 141: Pyramid chart with 0.1 GapRatio*

#### See Also

[Pyramid Chart](#), [Funnel Chart](#)

#### 4.5.1.32 Gradient

Gets / sets ColorBlend for a pie item. ColorBlend defines arrays of colors and positions used for interpolating color blending in a multicolor gradient.

Details	
<b>Possible Values</b>	A ColorBlend object
<b>Default Value</b>	<b>None</b>

<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Pie Chart

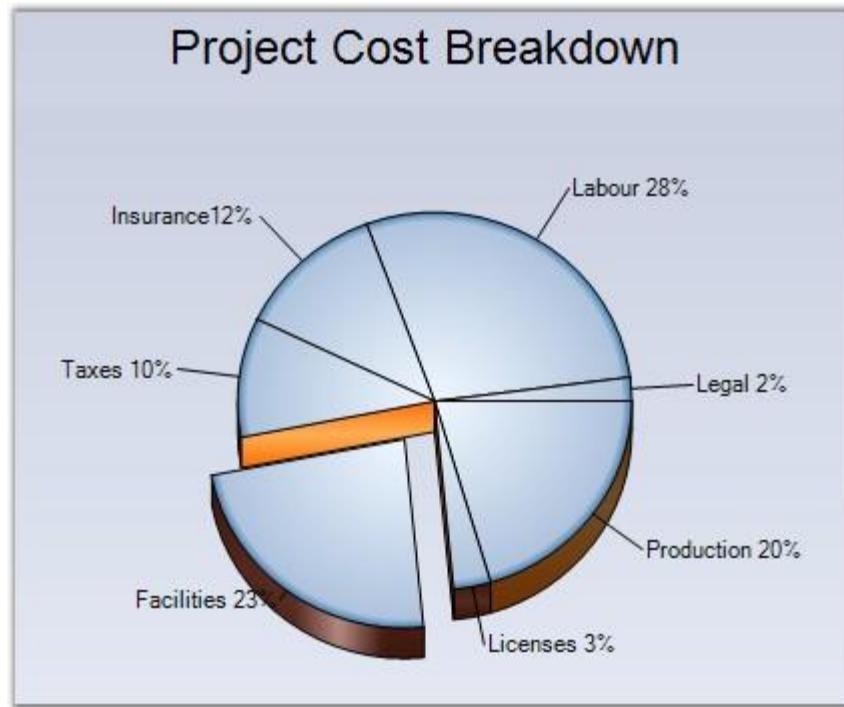
Here is some sample code.

**[C#]**

```
series.ConfigItems.PieItem.PieType = ChartPieType.Custom;
ColorBlend clrlnd = new ColorBlend();
clrlnd.Positions = new float[] { 0f, 0.05f, 1f };
clrlnd.Colors = new Color[] { Color.SteelBlue, Color.LightSteelBlue,
Color.AliceBlue };
// Specifying Gradient Style
series.ConfigItems.PieItem.Gradient = clrlnd;
```

**[VB .NET]**

```
series.ConfigItems.PieItem.PieType = ChartPieType.Custom
Private clrlnd As ColorBlend = New ColorBlend()
clrlnd.Positions = New Single() { 0f, 0.05f, 1f }
clrlnd.Colors = New Color() { Color.SteelBlue, Color.LightSteelBlue,
Color.AliceBlue }
' Specifying Gradient Style
series.ConfigItems.PieItem.Gradient = clrlnd
```



*Figure 142: Pie Chart with Gradient Styles Set*

#### **See Also**

[Pie Chart](#)

#### **4.5.1.33 HeightBox**

Gets / sets the height of the boxes in the financial chart types.

Details	
<b>Possible Values</b>	A double value
<b>Default Value</b>	1
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Point And Figure Chart

Here is some sample code.

**[C#]**

```
this.chartControl1.Series[0].HeightBox = 2f;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).HeightBox = 2f
```



Figure 143: PointAndFigure Chart with Box Size of 1 (Default)



Figure 144: PointAndFigure Chart with Box Size of 2

### See Also

[PointAndFigure Chart](#)

#### 4.5.1.34 HeightByAreaDepth

Indicates whether to draw series using the **ChartArea.Depth** property.

Details	
<b>Possible Values</b>	True or False
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	3D Only
<b>Applies to Chart Element</b>	All series

<b>Applies to Chart Types</b>	Pie Chart
-------------------------------	-----------

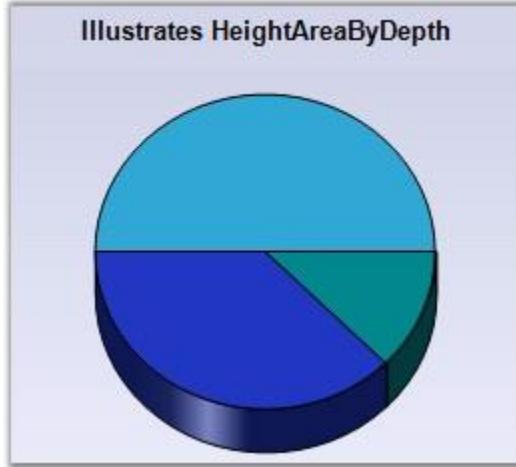
Here is some sample code.

**[C#]**

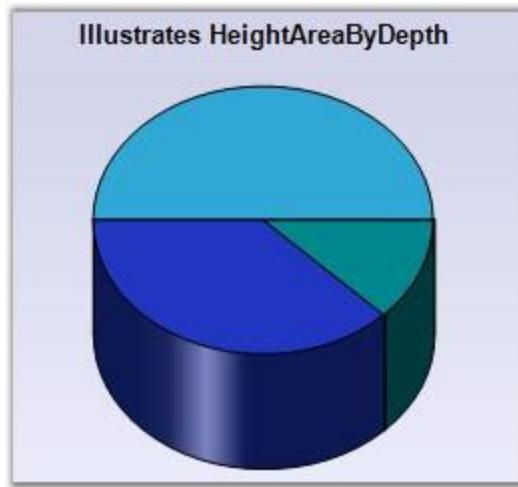
```
this.chartControl1.Series[0].ConfigItems.PieItem.HeightByAreaDepth =  
true;  
this.chartControl1.ChartArea.Depth = 25f;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.PieItem.HeightByAreaDepth = True  
Me.chartControl1.ChartArea.Depth = 25f
```



*Figure 145: Pie Chart with Default Dimension*



*Figure 146: Pie Chart with HeightByAreaDepth Enabled*

#### **See Also**

[Pie Chart](#)

#### **4.5.1.35 HeightCoefficient**

When in **3D** mode, the relative height of the pie chart can be specified via the property. Note that the **HeightByAreaDepth** property should be set as **false** for this to take effect.

Details	
<b>Possible Values</b>	Valid Ranges From 0 to 1
<b>Default Value</b>	<b>0.2</b>
<b>2D / 3D Limitations</b>	3D Only
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Pie Chart

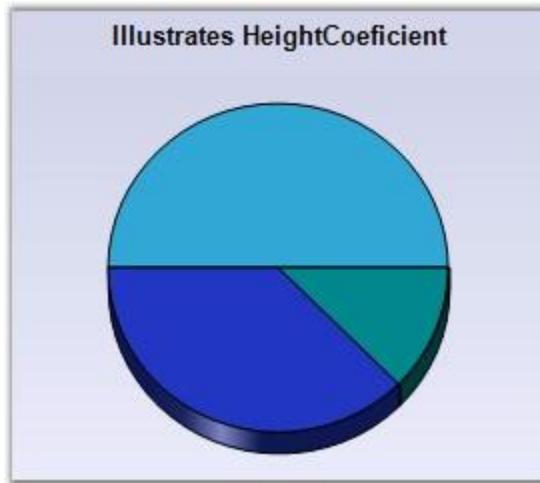
Here is the sample code.

[C#]

```
this.chartControl1.Series[0].ConfigItems.PieItem.HeightByAreaDepth =  
false;  
this.chartControl1.Series[0].ConfigItems.PieItem.HeightCoeficient =  
0.1f;
```

[VB.NET]

```
Me.chartControl1.Series(0).ConfigItems.PieItem.HeightByAreaDepth =  
False  
Me.chartControl1.Series(0).ConfigItems.PieItem.HeightCoeficient=0.1f
```



*Figure 147: Pie Chart with HeightCoeficient*

**See Also**

[Pie Chart](#)

#### 4.5.1.36 HighlightInterior

The auto highlight color for any series can be changed by setting the color at the **HighlightInterior** property of **ChartStyleInfo** class.

**Details**

<b>Possible Values</b>	RGB values ranges from 0 to 255
<b>Default Value</b>	<b>None</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Bar Charts, Pie, Funnel, Pyramid,Bubble, Column, Area, Stacking Area, Stacking Area100, Line Charts, Box and Whisker, Gantt Chart , Tornado Chart, Polar And Radar Chart, Hi Lo Chart, Hi Lo Open Close Chart

Here is some sample code.

### **Series Wide Setting**

**[C#]**

```
this.chartControl1.AutoHighlight = true;
ChartSeries series1 = this.chartControl1.Series[0];
series1.Style.HighlightInterior = new
BrushInfo(GradientStyle.ForwardDiagonal, Color.Red, Color.White);
```

**[VB .NET]**

```
Me.chartControl1.AutoHighlight = True
Dim series1 As ChartSeries = Me.chartControl1.Series(0)
series1.Style.HighlightInterior = New
BrushInfo(GradientStyle.ForwardDiagonal, Color.Red, Color.White)
```

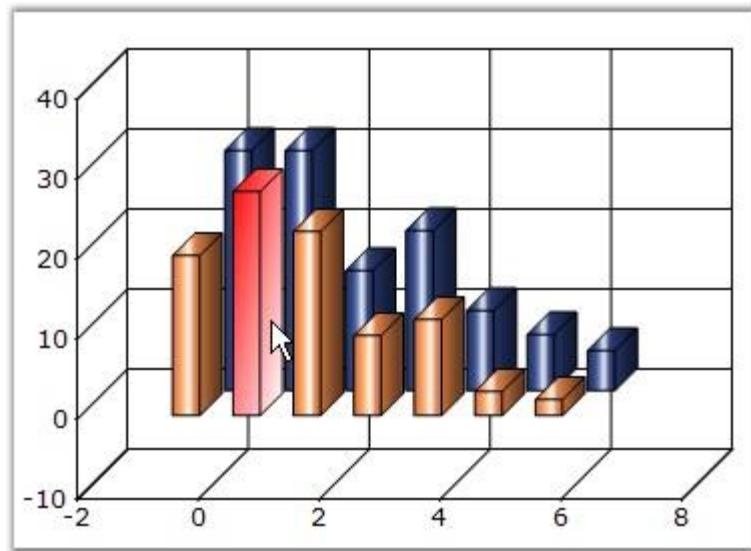


Figure 148: Column Chart with HighlightInterior

### Specific Data Point Setting

To set interior color for individual highlighted datapoints,

#### [C#]

```
series1.Styles[0].HighlightInterior = new  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Red, Color.White);  
series1.Styles[1].HighlightInterior = new  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Green, Color.White);  
series1.Styles[2].HighlightInterior = new  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Yellow, Color.White);  
series1.Styles[3].HighlightInterior = new  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Pink, Color.White);
```

#### [VB.NET]

```
series1.Styles(0).HighlightInterior = New  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Red, Color.White)  
series1.Styles(1).HighlightInterior = New  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Green, Color.White)  
series1.Styles(2).HighlightInterior = New  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Yellow, Color.White)  
series1.Styles(3).HighlightInterior = New  
BrushInfo(GradientStyle.ForwardDiagonal, Color.Pink, Color.White)
```

### See Also

[Bar Charts](#), [Pie Chart](#), [Funnel Chart](#), [Pyramid Chart](#), [Bubble Chart](#), [Column Chart](#), [Area Chart](#), [Stacking Area Chart](#), [Stacking Area100 Chart](#), [Line Charts](#), [Box and Whisker Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Polar And Radar Chart](#), [Hi Lo Chart](#), [Hi Lo Open Close Chart](#)

#### 4.5.1.37 HitTestRadius

HitTestRadius property controls the circle around this point, which will be considered within the bounds of this point for hit-testing purposes. The ChartRegion events such as ChartRegionClick, ChartRegionMouseDown, ChartRegionMouseHover, ChartRegionMouseLeave, ChartRegionMouseMove and ChartRegionMouseEnter, are being affected by this property.

Details	
<b>Possible Values</b>	A double values
<b>Default Value</b>	7.5
<b>2D / 3D Limitations</b>	None
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Line Chart and Step Line Chart

Here is some sample code.

[C#]

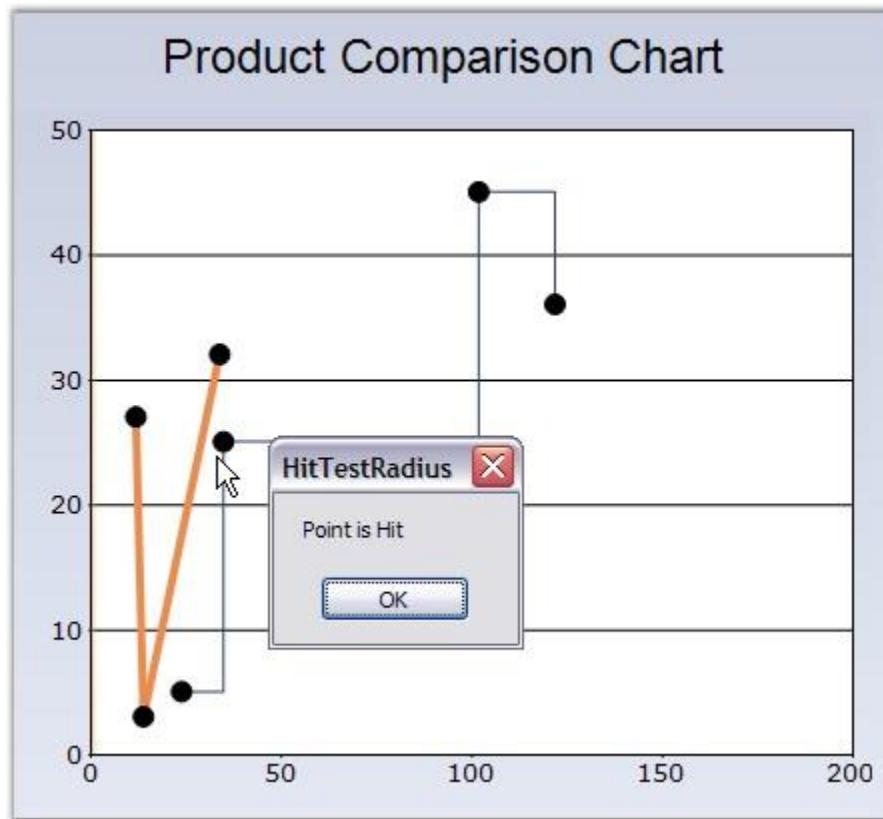
```
// Specifies the circle radius around the point for HitTest
this.chartControl1.Series[0].Style.HitTestRadius = 20;

// ChartClick Event will be fired if clicked within the above circle
void chartControl1_ChartRegionClick(object sender,
Syncfusion.Windows.Forms.Chart.ChartRegionEventArgs e)
{
    // Message appears when User hits the test radius region
    if (e.Region.IsChartPoint)
    {
        MessageBox.Show("Point is Hit");
    }
}
```

**[VB .NET]**

```
' Specifies the circle radius around the point for HitTest
Me.chartControl1.Series(0).Style.HitTestRadius = 20

' ChartClick Event will be fired if clicked within the above circle
Private Sub chartControl1_ChartRegionClick(ByVal sender As Object,
ByVal e As Syncfusion.Windows.Forms.Chart.ChartRegionEventArgs)
    ' Message appears when User hits the test radius region
    If e.Region.IsChartPoint Then
        MessageBox.Show("Point is Hit")
    End If
End Sub
```



*Figure 149: Chart with HitTestRadius = "20"*

**See Also**

[Line Chart](#) , [StepLineChart](#), [ChartRegionClick Events](#)

#### 4.5.1.38 ImageIndex

Gets / sets the image index from the associated **ImageList** property.

Details	
<b>Possible Values</b>	A numeric value indicating an index of the image list.
<b>Default Value</b>	<b>None</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	Area Charts, Bar Charts, Bubble Chart, Column Charts, Line Charts, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Tornado Chart, Polar and Radar Chart

Here is some sample code.

#### Series Wide Setting

[C#]

```
// Setting Images For the Series1
series1.Style.Images = new
ChartImageCollection(this.imageList1.Images);
series1.Style.Symbol.ImageIndex = 0;
series1.Style.Symbol.Size = new Size(20, 20);
series1.Style.Symbol.Shape = ChartSymbolShape.Image;
```

[VB .NET]

```
' Setting Images For the Series1
series1.Style.Images = New ChartImageCollection(Me.imageList1.Images)
series1.Style.Symbol.ImageIndex = 0
series1.Style.Symbol.Size = New Size(20, 20)
series1.Style.Symbol.Shape = ChartSymbolShape.Image
```

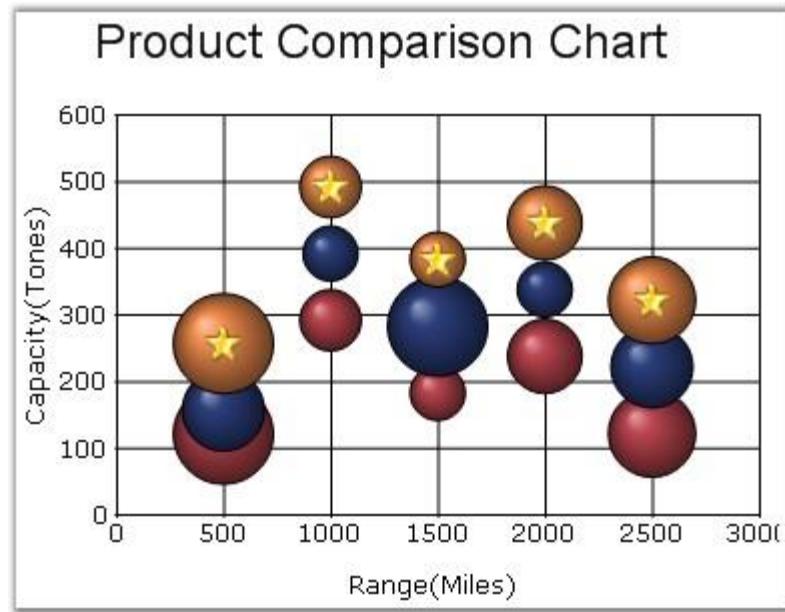


Figure 150: Bubble Chart

#### Specific Data Point Setting

##### [C#]

```
//Symbol set for specific data points (first point)
series1Styles[0].Images = new
ChartImageCollection(this.imageList1.Images);
series1Styles[0].Symbol.ImageIndex = 0;
series1Styles[0].Symbol.Size = new Size(20, 20);
series1Styles[0].Symbol.Shape = ChartSymbolShape.Image;

//Symbol set for specific data points (Second point)
series1Styles[1].Images = new
ChartImageCollection(this.imageList1.Images);
series1Styles[1].Symbol.ImageIndex = 1;
series1Styles[1].Symbol.Size = new Size(20, 20);
series1Styles[1].Symbol.Shape = ChartSymbolShape.Image;
```

##### [VB .NET]

```
'Symbol set for specific data points (first point )
series1Styles(0).Images = New
ChartImageCollection(Me.imageList1.Images)
series1Styles(0).Symbol.ImageIndex = 0
series1Styles(0).Symbol.Size = New Size(20, 20)
```

```

series1.Styles(0).Symbol.Shape = ChartSymbolShape.Image

//Symbol set for specific data points (Second point here)
series1.Styles(1).Images = New
ChartImageCollection(Me.imageList1.Images)
series1.Styles(1).Symbol.ImageIndex = 1
series1.Styles(1).Symbol.Size = New Size(20, 20)
series1.Styles(1).Symbol.Shape = ChartSymbolShape.Image

```

### See Also

[Area Charts](#), [Bar Charts](#), [Bubble Chart](#), [Column Charts](#), [Line Charts](#), [Candle Chart](#), [Renko chart](#), [Three Line Break Chart](#), [Box and Whisker Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Polar and Radar Chart](#)

### 4.5.1.39 Images

Gets / sets the imagelist that is to be associated with this ChartPoint. This property is used in conjunction with the **ImageIndex** property to display images associated with this point.

Details	
<b>Possible Values</b>	Value that represents the custom ImageList.
<b>Default Value</b>	<b>None</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	Area Charts, Bar Charts, Bubble Chart, Column Charts, Line Charts, Candle Chart, Renko chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Tornado Chart, Polar and Radar Chart

Here is some sample code.

```

[C#]

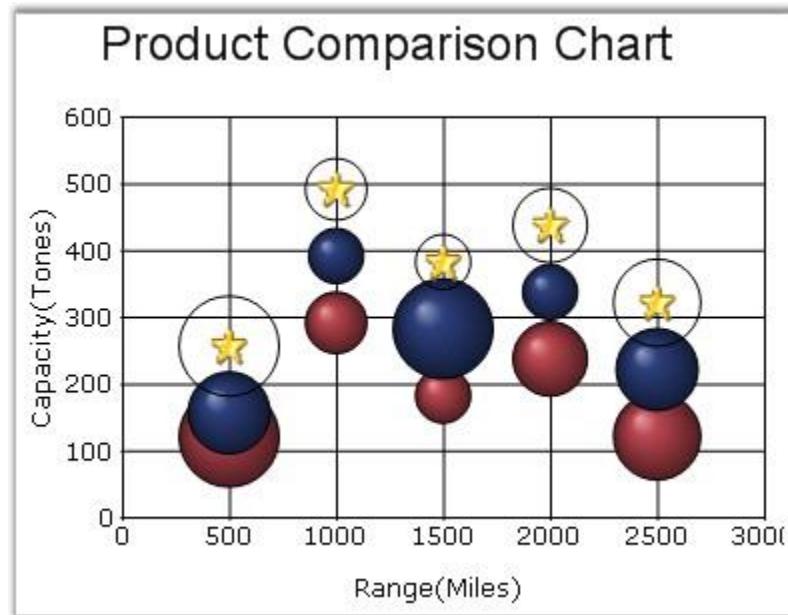
// Setting Images For the Series1

```

```
series1.Style.Images = new  
ChartImageCollection(this.imageList1.Images);  
series1.Style.Symbol.ImageIndex = 0;  
series1.Style.Symbol.Size = new Size(20, 20);  
series1.Style.Symbol.Shape = ChartSymbolShape.Image;  
  
// Disabling PhongStyle  
this.chartControl1.Series[0].ConfigItems.BubbleItem.EnablePhongStyle =  
false;
```

**[VB.NET]**

```
' Setting Images For the Series1  
series1.Style.Images = New ChartImageCollection(Me.imageList1.Images)  
series1.Style.Symbol.ImageIndex = 0  
series1.Style.Symbol.Size = New Size(20, 20)  
series1.Style.Symbol.Shape = ChartSymbolShape.Image  
  
' Disabling PhongStyle  
Me.chartControl1.Series(0).ConfigItems.BubbleItem.EnablePhongStyle =  
False
```



*Figure 151: Bubble Chart with Image*

#### **Specific Data Point Setting**

You can also specify different image collections for different data points using the below code.

**[C#]**

```
series1.Styles[0].Images = new
ChartImageCollection(this.imageList1.Images);
series1.Styles[0].Symbol.ImageIndex = 1;
series1.Styles[0].Symbol.Size = new Size(20, 20);
series1.Styles[0].Symbol.Shape = ChartSymbolShape.Image;

series1.Styles[1].Images = new
ChartImageCollection(this.imageList2.Images);
series1.Styles[1].Symbol.ImageIndex = 2;
series1.Styles[1].Symbol.Size = new Size(20, 20);
series1.Styles[1].Symbol.Shape = ChartSymbolShape.Image;
```

**[VB.NET]**

```
series1.Styles(0).Images = New
ChartImageCollection(Me.imageList1.Images)
series1.Styles(0).Symbol.ImageIndex = 1
series1.Styles(0).Symbol.Size = New Size(20, 20)
series1.Styles(0).Symbol.Shape = ChartSymbolShape.Image

series1.Styles(1).Images = New
ChartImageCollection(Me.imageList2.Images)
series1.Styles(1).Symbol.ImageIndex = 2
series1.Styles(1).Symbol.Size = New Size(20, 20)
series1.Styles(1).Symbol.Shape = ChartSymbolShape.Image
```

**See Also**

[Area Charts](#), [Bar Charts](#), [Bubble Chart](#), [Column Charts](#), [Line Charts](#), [Candle Chart](#), [Renko chart](#), [Three Line Break Chart](#), [Box and Whisker Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Polar and Radar Chart](#)

**4.5.1.40 InnerRadius**

Sets / Gets the radius of the doughnut hole of Pie chart as a fraction of the radius of the pie.

Details	
Possible Values	Ranges from 0.0f to 1.0f.

<b>Default Value</b>	None.
<b>2D / 3D Limitations</b>	No.
<b>Applies to Chart Element</b>	All series.
<b>Applies to Chart Types</b>	Pie Chart.

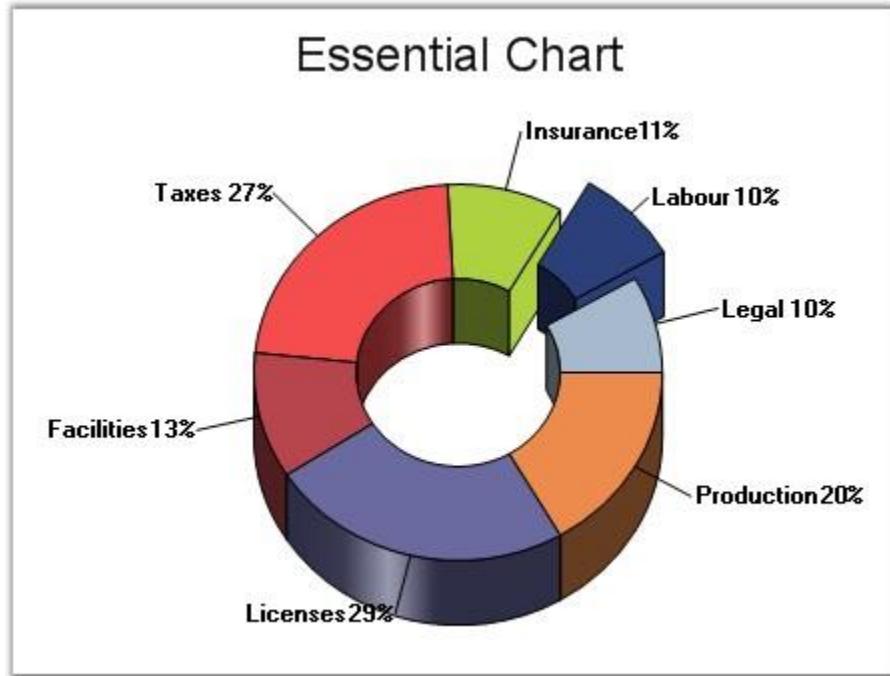
Here is some sample code.

[C#]

```
ChartSeries series1 = this.chartControl1.Model.NewSeries("Market");
series1.InnerRadius = 0.5f;
```

[VB .NET]

```
Dim series1 As ChartSeries = Me.chartControl1.Model.NewSeries("Market")
series1.InnerRadius = 0.5f
```



*Figure 152: InnerRadius Pie Chart*

## See Also

[Pie Chart](#)

#### 4.5.1.41 Interior

This property will allow the user to set a solid back color, gradient or pattern style for the data points.

Details	
<b>Possible Values</b>	A BrushInfo object
<b>Default Value</b>	<b>None</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series and points
<b>Applies to Chart Types</b>	All Chart Types

##### Series Wide Setting

The spline area interior brush can be customized using the **ChartSeries.Style.Interior** property as shown below.

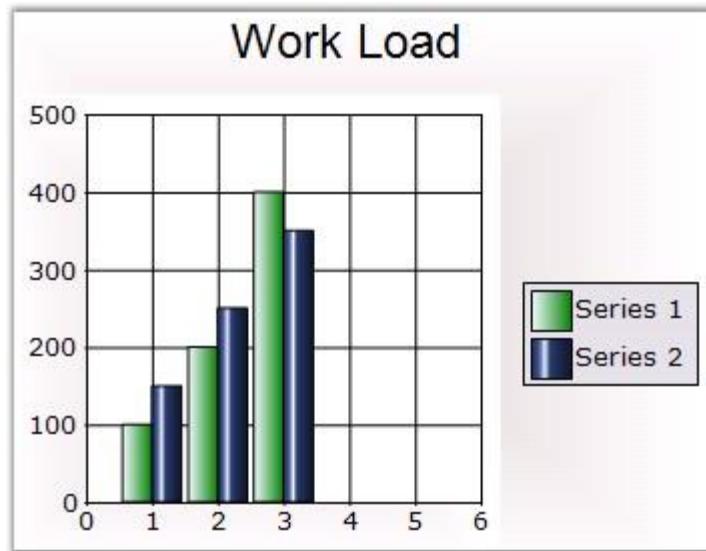
The interior color of the chart series can be customized by using the **Interior** property of the **ChartStyleInfo** class. The following code illustrates this.

**[C#]**

```
// This sets the interior color for the series. This can be done for
any number of series.
this.chartControl1.Series[0].Style.Interior = new
BrushInfo(GradientStyle.Horizontal ,Color.AliceBlue, Color.Green);
```

**[VB .NET]**

```
' This sets the interior color for the series. This can be done for any
number of series.
Me.chartControl1.Series(0).Style.Interior = New
BrushInfo(GradientStyle.Horizontal,Color.AliceBlue, Color.Green)
```



*Figure 153: Chart control with Gradient Control*

### Specific Data Point Setting

You can also set interior color for individual data points using **Series.Styles[0].Interior** property.

[C#]

```
this.chartControl1.Series[0].Styles[0].Interior = new  
BrushInfo(GradientStyle.Horizontal, Color.AliceBlue, Color.Green);  
this.chartControl1.Series[0].Styles[1].Interior = new  
BrushInfo(GradientStyle.Horizontal, Color.Blue, Color.AliceBlue);
```

[VB .NET]

```
Me.chartControl1.Series(0).Styles[0].Interior = New  
BrushInfo(GradientStyle.Horizontal, Color.AliceBlue, Color.Green)  
Me.chartControl1.Series(0).Styles[1].Interior = New  
BrushInfo(GradientStyle.Horizontal, Color.Blue, Color.AliceBlue)
```

### PieChart Specific

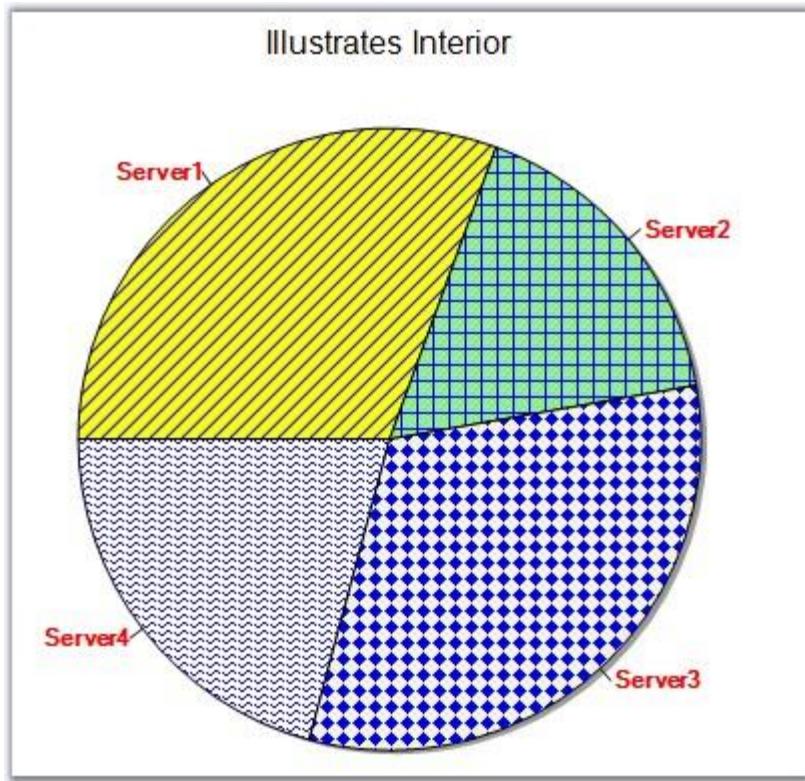
When rendering pie charts, it's sometimes very helpful to render a patterned background for each slice, while printing the pie on a gray scale printer. You can do so easily as shown below. The code here is for a Pie Chart series with 4 points.

[C#]

```
series1.Styles[0].Interior = new
BrushInfo(PatternStyle.BackwardDiagonal, new
BrushInfoColorArrayList(new Color[] { Color.Yellow, Color.Blue }));
series1.Styles[1].Interior = new BrushInfo(PatternStyle.Cross, new
BrushInfoColorArrayList(new Color[] { Color.LightGreen, Color.Blue }));
series1.Styles[2].Interior = new BrushInfo(PatternStyle.SolidDiamond,
new BrushInfoColorArrayList(new Color[] { Color.Beige, Color.Blue }));
series1.Styles[3].Interior = new BrushInfo(PatternStyle.Wave, new
BrushInfoColorArrayList(new Color[] { Color.White, Color.Blue }));
series1.Styles[0].Text = "Server1";
series1.Styles[1].Text = "Server2";
series1.Styles[2].Text = "Server3";
series1.Styles[3].Text = "Server4";
```

**[VB.NET]**

```
series1.Styles(0).Interior = New
BrushInfo(PatternStyle.BackwardDiagonal, New
BrushInfoColorArrayList(New Color() { Color.Yellow, Color.Blue }))
series1.Styles(1).Interior = New BrushInfo(PatternStyle.Cross, New
BrushInfoColorArrayList(New Color() { Color.LightGreen, Color.Blue }))
series1.Styles(2).Interior = New BrushInfo(PatternStyle.SolidDiamond,
New BrushInfoColorArrayList(New Color() { Color.Beige, Color.Blue }))
series1.Styles(3).Interior = New BrushInfo(PatternStyle.Wave, New
BrushInfoColorArrayList(New Color() { Color.White, Color.Blue }))
series1.Styles(0).Text = "Server1"
series1.Styles(1).Text = "Server2"
series1.Styles(2).Text = "Server3"
series1.Styles(3).Text = "Server4"
```



*Figure 154: Unique PatternStyle for each Slice in Pie Chart*

#### **See Also**

[Chart Types](#)

#### **4.5.1.42 LabelPlacement**

Gets or sets the Pyramid chart or Funnel chart data point label placement when **ChartAccumulationLabelStyle** is set as **Inside**.

Details	
Possible Values	
	<ul style="list-style-type: none"><li>• <b>Center</b> – DataPoint labels are aligned to the center of the Pyramid segment.</li><li>• <b>Top</b> - DataPoint labels are aligned to the top of the Pyramid segment.</li></ul>

	<ul style="list-style-type: none"><li>• <b>Bottom</b> – DataPoint labels are aligned to the bottom of the Pyramid segment.</li><li>• <b>Left</b> - DataPoint labels are aligned to the Left of the Pyramid segment.</li><li>• <b>Right</b> - DataPoint labels are aligned to the Right of the Pyramid segment.</li></ul>
<b>Default Value</b>	Right
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Funnel and Pyramid Charts

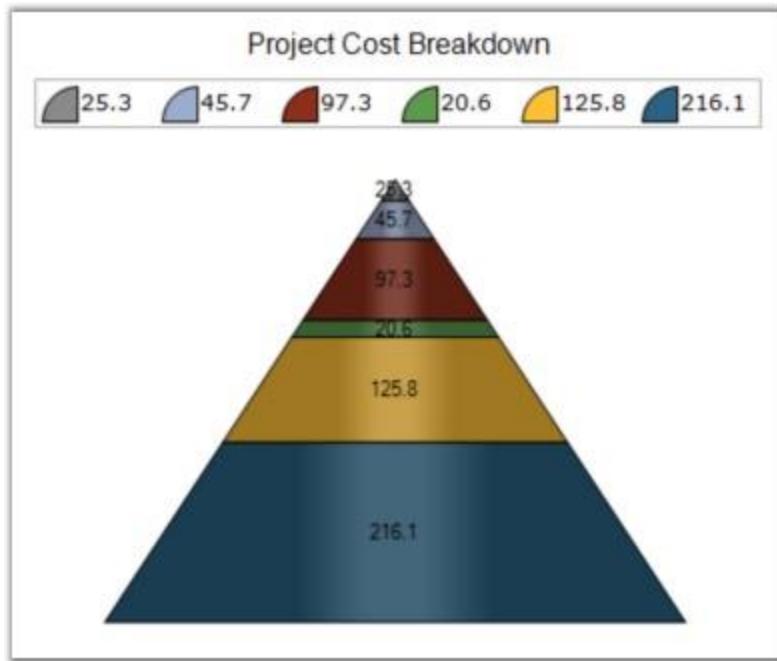
Here is the code snippet using LabelPlacement in Pyramid Chart.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.PyramidItem.LabelPlacement =  
ChartAccumulationLabelPlacement.Center;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.PyramidItem.LabelPlacement =  
ChartAccumulationLabelPlacement.Center
```



*Figure 155: ChartAccumulationLabelPlacement as Center*

Here is the code snippet using LabelPlacement in Funnel Chart.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.FunnelItem.LabelPlacement =  
ChartAccumulationLabelPlacement.Center;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.FunnelItem.LabelPlacement =  
ChartAccumulationLabelPlacement.Center
```

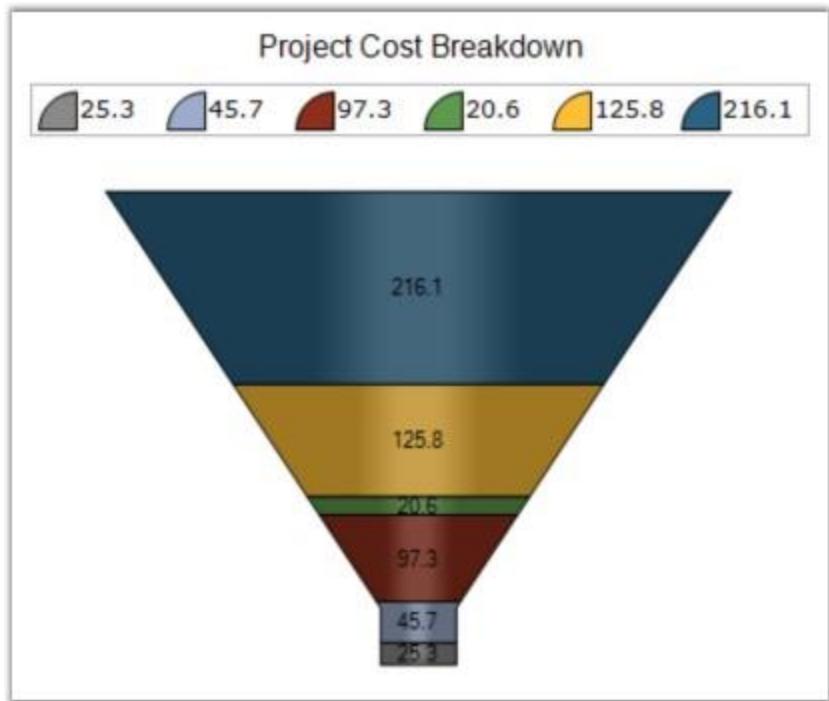


Figure 156: ChartAccumulationLabelPlacement as Center

## See Also

[Pyramid Chart](#), [Funnel Chart](#)

### 4.5.1.43 LabelStyle

Gets or sets the Pyramid chart or Funnel chart label style. This property dictates the presence and overall positioning of the DataPoint labels.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li><b>Inside</b> - DataPoint label is drawn inside the Pyramid segment.</li> <li><b>Outside</b> - DataPoint label is drawn outside and next to the Pyramid segment.</li> <li><b>OutsideInColumn</b> - DataPoint is drawn outside the Pyramid segment with all labels aligned.</li> </ul>

	<ul style="list-style-type: none"> <li>• <b>Disabled</b> - DataPoint labels are disabled.</li> </ul>
<b>Default Value</b>	<b>OutsideInColumn</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Funnel, Pyramid charts

Here is the code snippet using LabelStyle in Pyramid Chart.

[C#]

```
this.chartControl1.Series[0].ConfigItems.PyramidItem.LabelStyle =
ChartAccumulationLabelStyle.OutsideInColumn;
```

[VB .NET]

```
Me.chartControl1.Series(0).ConfigItems.PyramidItem.LabelStyle=
ChartAccumulationLabelStyle.OutsideInColumn
```

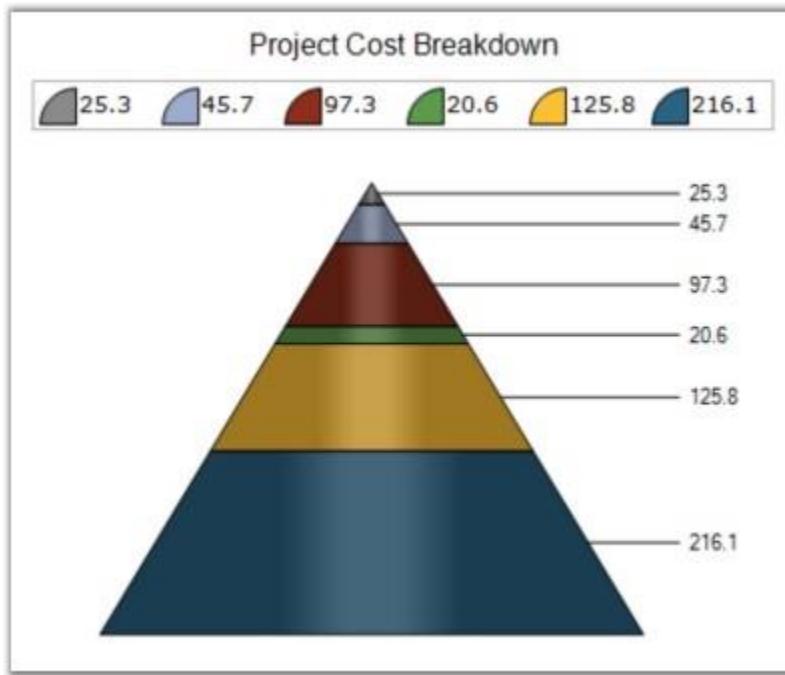


Figure 157: ChartAccumulationLabelStyle as OutsideInColumn

Here is the code snippet using LabelStyle in Funnel Chart.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.FunnelItem.LabelStyle =  
ChartAccumulationLabelStyle.OutsideInColumn;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.FunnelItem.LabelStyle=  
ChartAccumulationLabelStyle.OutsideInColumn
```

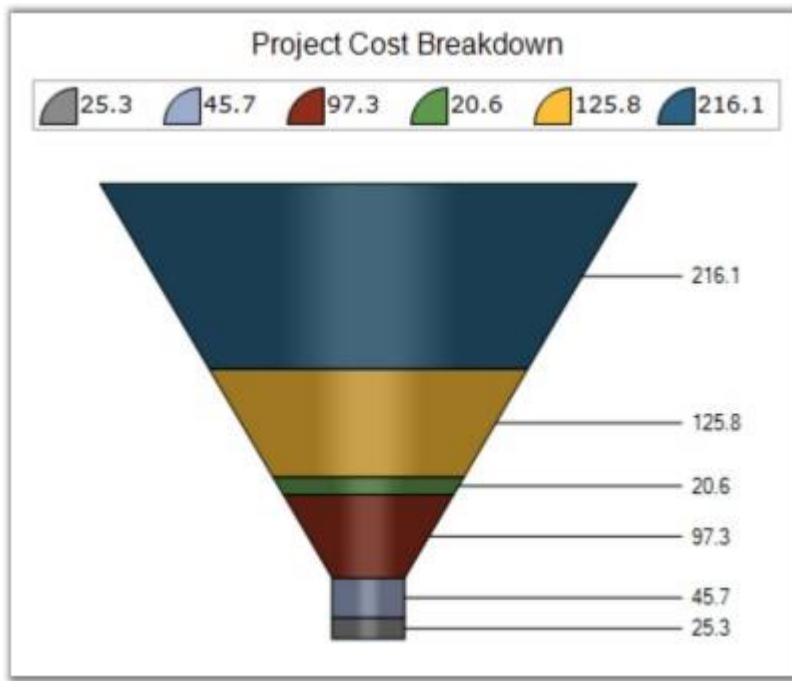


Figure 158: ChartAccumulationLabelStyle as OutsideInColumn

**See Also**

[Pyramid Chart](#), [Funnel Chart](#)

#### 4.5.1.44 LegendItem

Refer [ChartLegendItem](#) for more information.

#### 4.5.1.45 LightAngle

Specifies the light angle in horizontal plane.

Details	
<b>Possible Values</b>	Any double value
<b>Default Value</b>	<b>-0.785398163397448</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Column Charts , Bar Charts, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Polar and Radar Chart, Candle Chart, Hilo Chart(3D), HiloOpenClose(3D)

Here is code snippet using **LightAngle** in Column Chart.

**[C#]**

```
// Specifies light angle of both the series
this.chartControl1.Series[0].ConfigItems.ColumnItem.LightAngle = 45;
this.chartControl1.Series[1].ConfigItems.ColumnItem.LightAngle = 45;
```

**[VB .NET]**

```
' Specifies light angle of both the series
Private Me.chartControl1.Series(0).ConfigItems.ColumnItem.LightAngle
=45
Private Me.chartControl1.Series(1).ConfigItems.ColumnItem.LightAngle =
45
```



*Figure 159: Light Angle = 45 with 3D Effect*



*Figure 160: Light Angle = 30 with 3D Effect*

#### **See Also**

[Column Charts](#),[Bar Charts](#), [Box and Whisker Chart](#) , [Gantt Chart](#) , [Histogram Chart](#), [Tornado Chart](#) , [Polar and Radar Chart](#), [Candle Chart](#), [Hilo Chart\(3D\)](#), [HiloOpenClose\(3D\)](#)

#### 4.5.1.46 LightColor

Specifies the color of light for all shading modes except ChartColumnShadingMode.FlatRectangle.

Details	
<b>Possible Values</b>	A Color object
<b>Default Value</b>	<b>Color.White</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Column Chart, Bar Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Radar Chart

Here is sample code snippet using **LightColor** in Column Chart.

[C#]

```
this.chartControl1.Series[0].ConfigItems.ColumnItem.LightColor =
Color.Blue;
this.chartControl1.Series[1].ConfigItems.ColumnItem.LightColor =
Color.Green;
```

[VB .NET]

```
Private Me.chartControl1.Series(0).ConfigItems.ColumnItem.LightColor =
Color.Blue
Private Me.chartControl1.Series(1).ConfigItems.ColumnItem.LightColor =
Color.Green
```

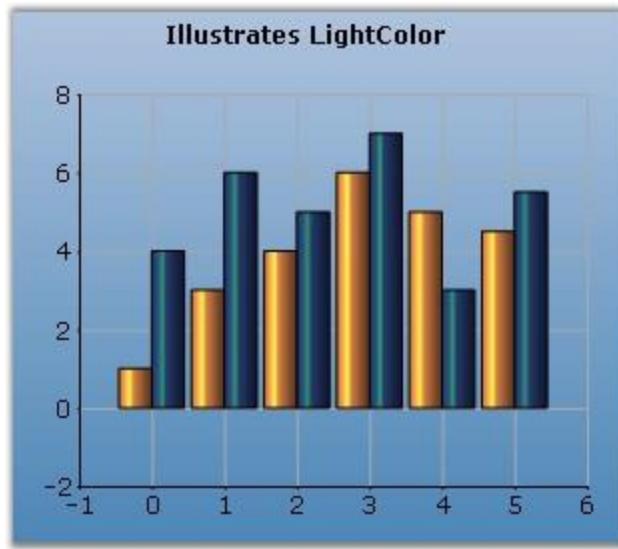


Figure 161: LightColor applied to Chart Series

#### See Also

[Scatter Chart](#), [Column Charts](#), [Bar Charts](#), [Box and Whisker Chart](#) , [Gantt Chart](#) , [Histogram Chart](#), [Tornado Chart](#) ,  
[Polar and Radar Chart](#)

#### 4.5.1.47 Name

Specifies the name of the Series. This name can also be used to retrieve the series by name from the series collection.

Details	
Possible Values	Any user defined string
Default Value	Null
2D / 3D Limitations	No
Applies to Chart Element	Any Series points
Applies to Chart Types	All chart types

Here is the code snippet using **Name** in Column Chart.

**[C#]**

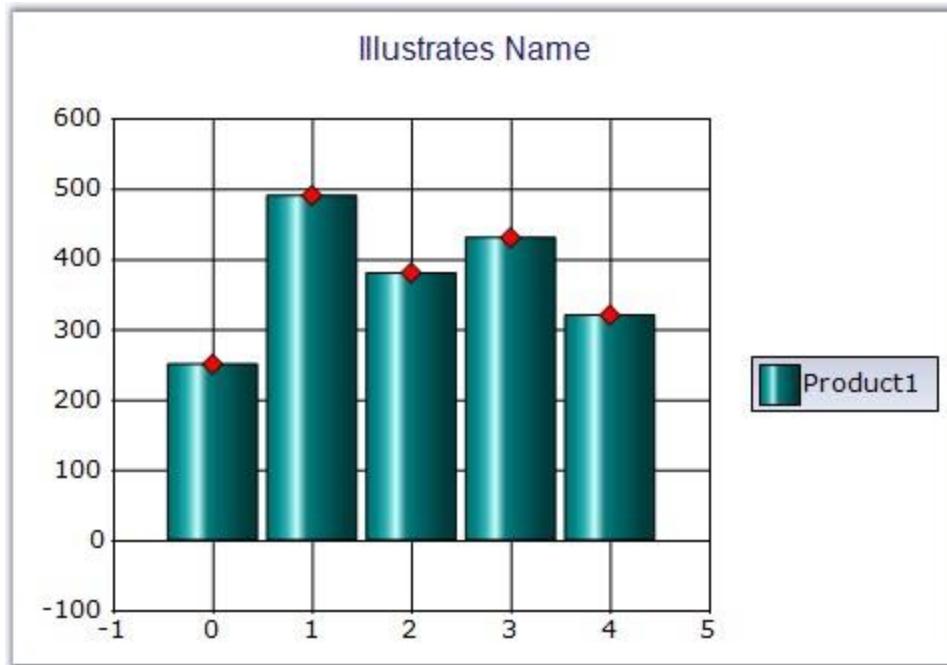
```
//This Code Snippet gives the name of the series as Product1
ChartSeries s1 = this.chartControl1.Model.NewSeries();
s1.Type = Syncfusion.Windows.Forms.Chart.ChartSeriesType.Column;
s1.Name="Product1";
// Points to be added
this.chartControl1.Series.Add(s1);

//Series retrieved using Name
this.chartControl1.Series["Product1"].Style.Symbol.Shape =
ChartSymbolShape.Diamond;
this.chartControl1.Series["Product1"].Style.Symbol.Color = Color.Red;
```

**[VB .NET]**

```
' This Code Snippet gives the name of the series as Product
s1 As ChartSeries = Me.chartControl1.Model.NewSeries()
s1.Type = Syncfusion.Windows.Forms.Chart.ChartSeriesType.Column
s1.Name="Product1"
' Points to be added
Me.chartControl1.Series.Add(s1)

'Series retrieved using Name
Me.chartControl1.Series["Product1"].Style.Symbol.Shape =
ChartSymbolShape.Diamond
Me.chartControl1.Series["Product1"].Style.Symbol.Color = Color.Red
```



*Figure 162: Chart with Legend for the Series*

#### See Also

[Chart Types](#)

#### 4.5.1.48 NumberOfHistogramIntervals

Gets or sets the number of Histogram intervals.

Details	
Possible Values	Any numeric value
Default Value	10
2D / 3D Limitations	No
Applies to Chart Element	All Series Points
Applies to Chart Types	HistoGram Chart

Here is a code sample.

**[C#]**

```
// Set the desired number of intervals required for the histogram chart.  
series.NumberOfHistogramIntervals = 20;
```

**[VB .NET]**

```
' Set the desired number of intervals required for the histogram chart.  
series.NumberOfHistogramIntervals = 20
```

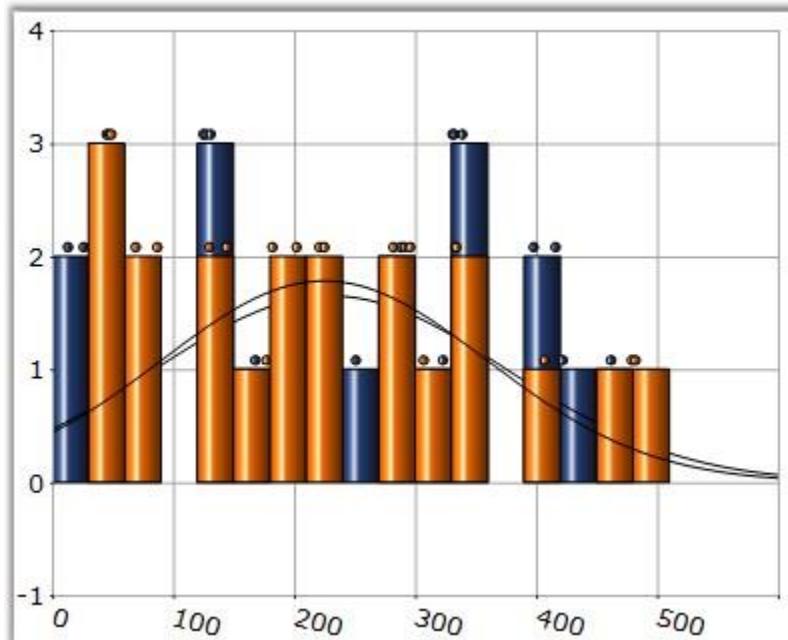


Figure 163: Histogram Chart with Interval set to 20

**See Also**

[Histogram Chart](#)

#### 4.5.1.49 OpenCloseDrawMode

Gets or sets the open, close draw mode to the HiloOpenClose chart.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• Open - Points only the opening value of that period.</li> <li>• Close - Points out the closing value of that period.</li> <li>• Both - Points out both the opening and the closing values of that period.</li> </ul>
<b>Default Value</b>	<b>Both</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series points
<b>Applies to Chart Types</b>	HiLoOpenClose Chart

Here is the code snippet using OpenCloseDrawMode.

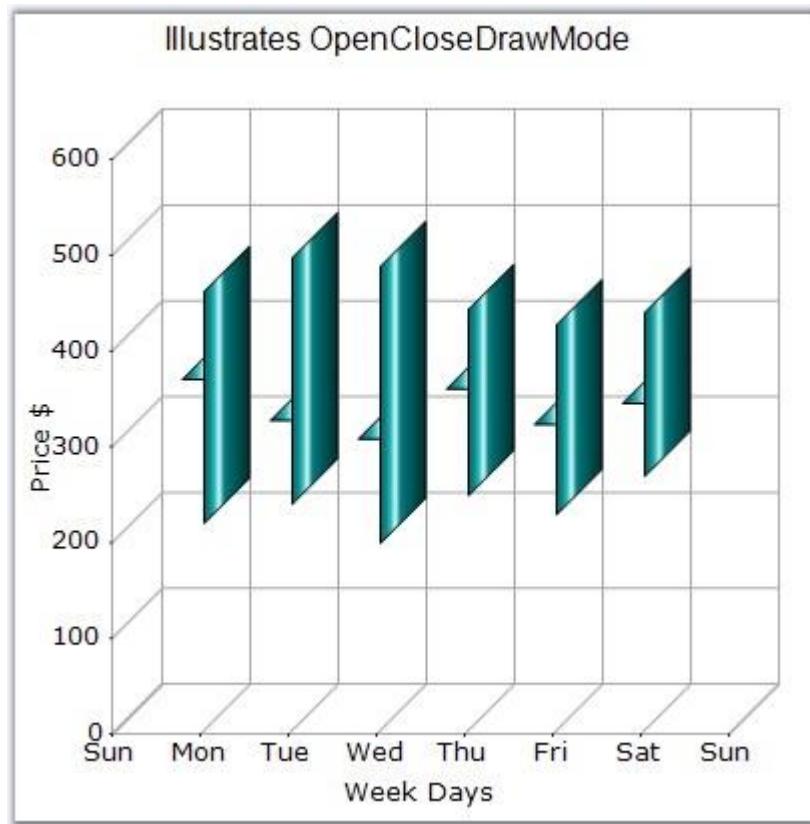
#### [C#]

```
ChartSeries CS1 = this.chartControl1.Model.NewSeries("Series
Name", ChartSeriesType.HiLoOpenClose);
CS1.Points.Add(date, 456, 214, 364, 386);
CS1.Points.Add(date.AddDays(1), 491, 234, 321, 378 );
CS1.Points.Add(date.AddDays(2), 482, 193, 302, 352);
CS1.Points.Add(date.AddDays(3), 437, 243, 354, 391);
CS1.Points.Add(date.AddDays(4), 421, 223, 317, 367);
CS1.Points.Add(date.AddDays(5), 434, 263, 339, 385);
this.chartControl1.Series.Add(CS1);
this.chartControl1.Series[0].OpenCloseDrawMode =
ChartOpenCloseDrawMode.Open;
```

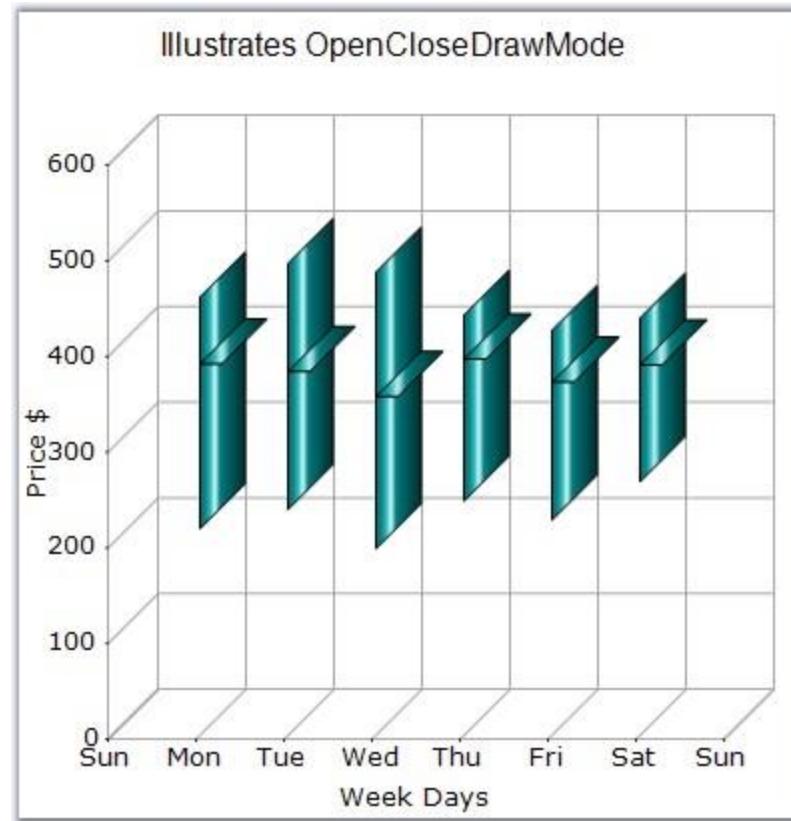
#### [VB .NET]

```
Dim CS1 As ChartSeries = Me.chartControl1.Model.NewSeries("Series
Name", ChartSeriesType.HiLoOpenClose)
CS1.Points.Add(date, 456, 214, 364, 386)
CS1.Points.Add(date.AddDays(1), 491, 234, 321, 378 )
CS1.Points.Add(date.AddDays(2), 482, 193, 302, 352)
CS1.Points.Add(date.AddDays(3), 437, 243, 354, 391)
CS1.Points.Add(date.AddDays(4), 421, 223, 317, 367)
CS1.Points.Add(date.AddDays(5), 434, 263, 339, 385)
```

```
Me.chartControl1.Series.Add(CS1)
Me.chartControl1.Series(0).OpenCloseDrawMode =
ChartOpenCloseDrawMode.Open
```



*Figure 164: Chart with "Open" Mode*



*Figure 165: Chart with "Close" Mode*

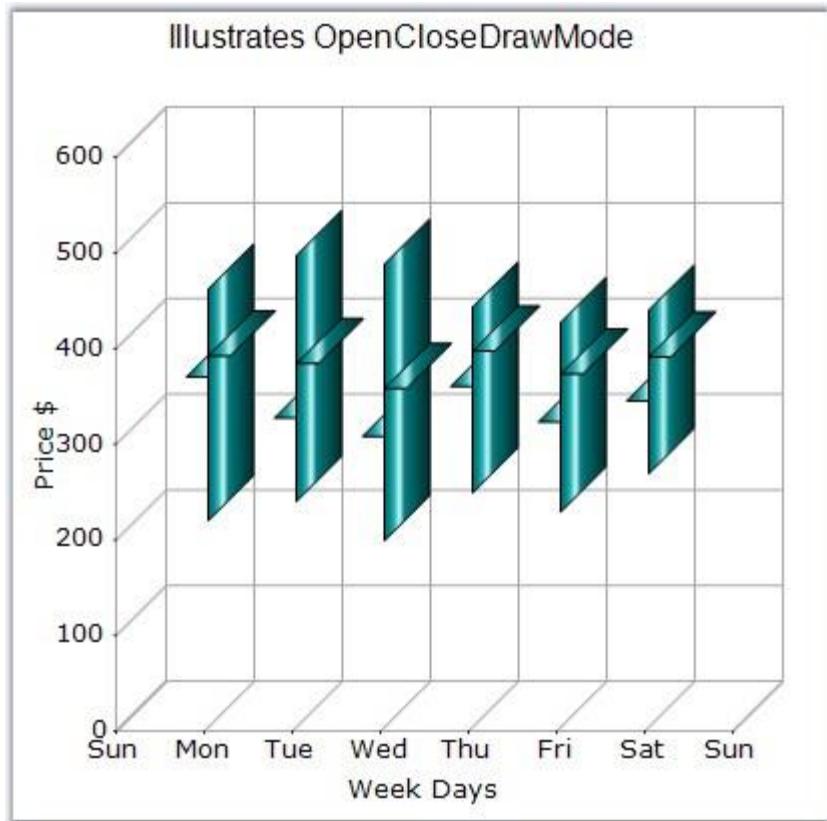


Figure 166: Chart with "Both" Mode

#### See Also

[Hi Lo Open Close Chart](#)

#### 4.5.1.50 OptimizePiePointPositions

Specifies if the data points with smaller values are grouped together and ordered. By default, they are ordered in the order in which the points are added to the series.

Details	
Possible Values	
	<ul style="list-style-type: none"><li>• True - Enables optimization</li><li>• False - Disables optimization</li></ul>

<b>Default Value</b>	<b>True</b>
<b>2D / 3D Limitations</b>	No.
<b>Applies to Chart Element</b>	Any Series.
<b>Applies to Chart Types</b>	Pie Chart.

Here is the code snippet using OptimizePiePointPositions.

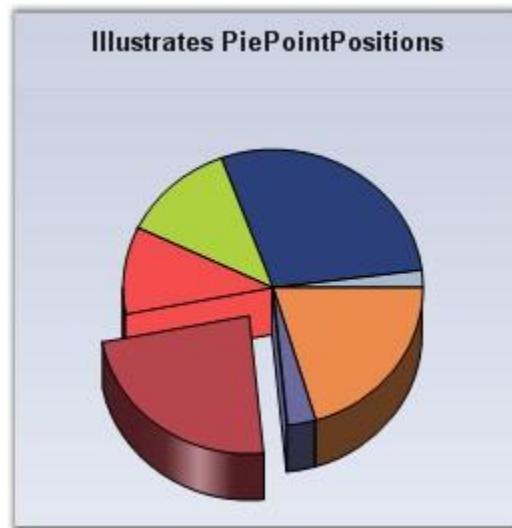
**[C#]**

```
ChartSeries series = this.chartControl1.Model.NewSeries("Series Name",
ChartSeriesType.Pie);
series.Points.Add(0, 20);
series.Points.Add(1, 28);
series.Points.Add(2, 23);
series.Points.Add(3, 10);
series.Points.Add(4, 12);
series.Points.Add(5, 3);
series.Points.Add(6, 2);
series.ExplodedIndex = 2;
series.OptimizePiePointPositions = false;
this.chartControl1.Series.Add(series);
```

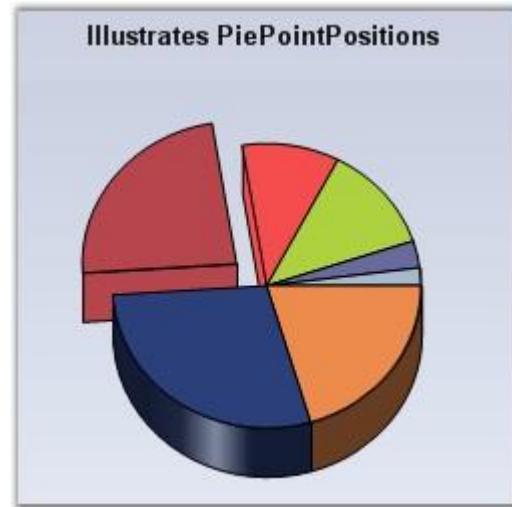
**[VB.NET]**

```
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Series
Name",ChartSeriesType.Pie)
series.Points.Add(0, 20)
series.Points.Add(1, 28)
series.Points.Add(2, 23)
series.Points.Add(3, 10)
series.Points.Add(4, 12)
series.Points.Add(5, 3)
series.Points.Add(6, 2)
series.ExplodedIndex = 2
series.OptimizePiePointPositions = False
```

```
Me.chartControll.Series.Add(series)
```



*Figure 167: OptimizePiePointPositions set to True*



*Figure 168: OptimizePiePointPositions set to False*

## See Also

[Pie Chart](#)

#### 4.5.1.51 PhongAlpha

Specifies the Phong's alpha co-efficient used for calculating specular lighting.

Details	
<b>Possible Values</b>	Any double value
<b>Default Value</b>	<b>20</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Column Chart, Bar Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Polar and Radar Chart, HiLo Chart, HiLoOpenClose Chart, Candle Chart, Scatter Chart

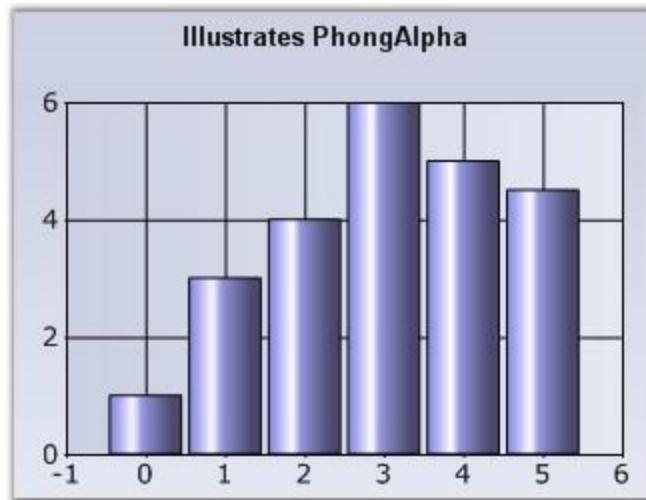
Here is code snippet using PhongAlpha in Column Chart.

[C#]

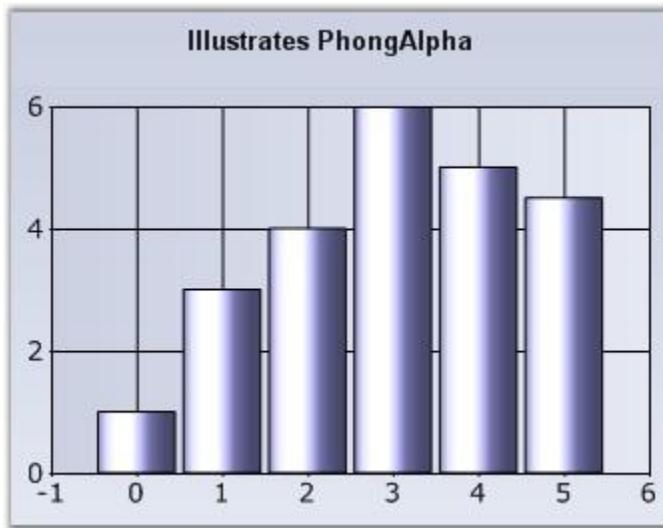
```
this.chartControl1.Series[0].ConfigItems.ColumnItem.PhongAlpha = 2.0;
```

[VB .NET]

```
Private Me.chartControl1.Series(0).ConfigItems.ColumnItem.PhongAlpha = 2.0
```



*Figure 169: Normal Column Chart*



*Figure 170: Column Chart with Phong Alpha Set*

#### **See Also**

[Bar Charts](#) , [Box and Whisker](#) , [Column Charts](#) , [Candle Chart](#) , [Gantt Chart](#) , [HiLo Chart](#) , [HiLoOpenClose Chart](#) , [Histogram Chart](#) , [Tornado Chart](#) , [Radar Chart](#) , [Scatter Chart](#),

### 4.5.1.52 PieType

Sets pre-defined types for pie charts.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• None - It has no specific type.</li> <li>• OutSide - Specifies outside pie type.</li> <li>• InSide - Specifies inside pie type.</li> <li>• Round - Specifies Round pie type.</li> <li>• Bevel - Specifies Bevel pie type.</li> <li>• Custom - Specifies custom pie type.</li> </ul>
<b>Default Value</b>	<b>None</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Pie Series
<b>Applies to Chart Types</b>	PieChart

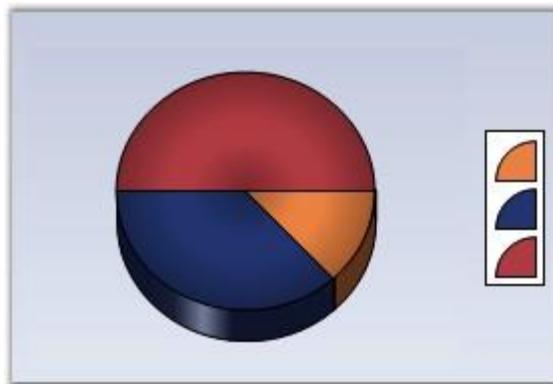
#### [C#]

```
this.chartControl1.Series[0].ConfigItems.PieItem.PieType=ChartPieType.Bevel;
```

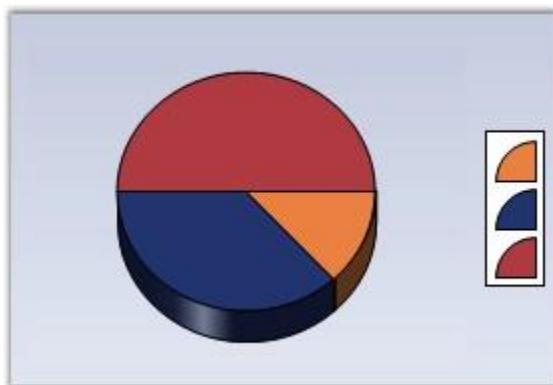
#### [VB .NET]

```
Me.chartControl1.Series(0).ConfigItems.PieItem.PieType =  
ChartPieType.Bevel
```

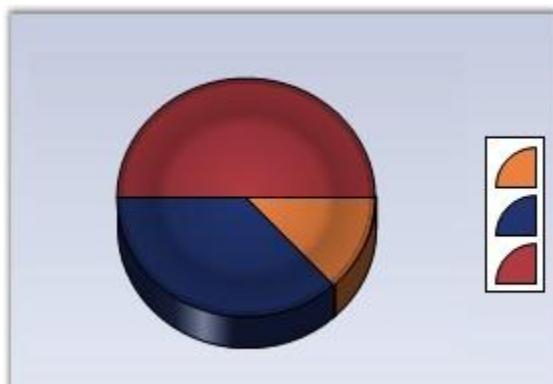
The following screen shots depict these types.



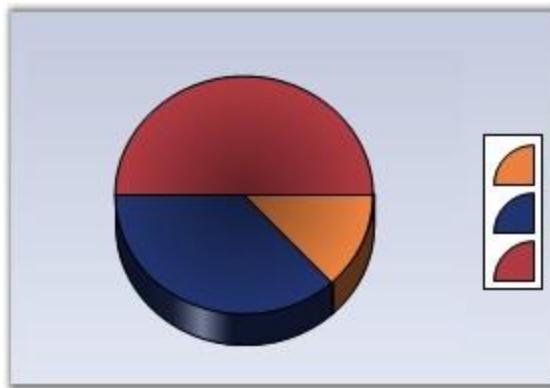
*Figure 171: Pie Chart*



*Figure 172: Pie Bevel Chart*



*Figure 173: Pie Inside Chart*



*Figure 174: Pie Outside Chart*



*Figure 175: Pie Round Chart*

#### **See Also**

[PieChart](#)

#### **4.5.1.53 PieWithSameRadius**

Gets or sets whether the pie chart is rendered in the same radius when the **LabelStyle** is set to **Outside** or **OutsideInColumn**.

Details	

<b>Possible Values</b>	True or False
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Pie Series
<b>Applies to Chart Types</b>	PieChart and Doughnut chart

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.PieItem.PieWithSameRadius=true  
;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).ConfigItems.PieItem.PieType = True
```

Setting this property to **true** will let you display Pie Chart with same size in the divided area.

#### 4.5.1.54 PointsToolTipFormat

Check [Tooltips](#) topic for more info on this setting.

##### See Also

[Chart Types](#)

#### 4.5.1.55 PointWidth

Sets the width of this point relative to the total width available. It is very useful to render series that overlap.

Details	
<b>Possible Values</b>	0.0F to 1.0F
<b>Default Value</b>	<b>1.0F</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Gantt Chart

Here is a code snippet using PointWidth in Gantt Chart.

### Series Wide Setting

[C#]

```
ganttSeries.Style.PointWidth = 0.25f;
```

[VB .NET]

```
Private ganttSeries.Style.PointWidth = 0.25f
```

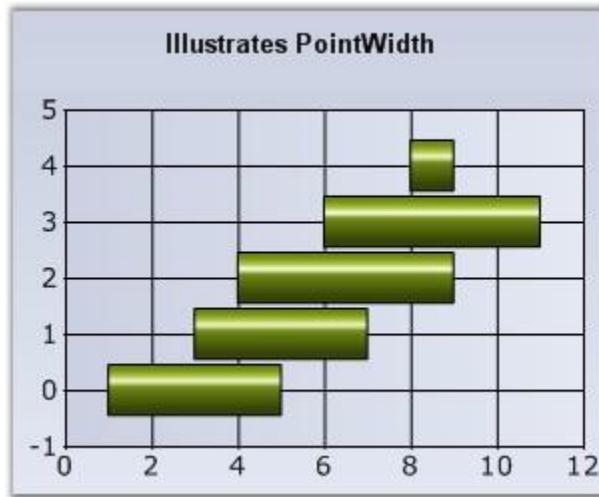


Figure 176: Chart with default PointWidth

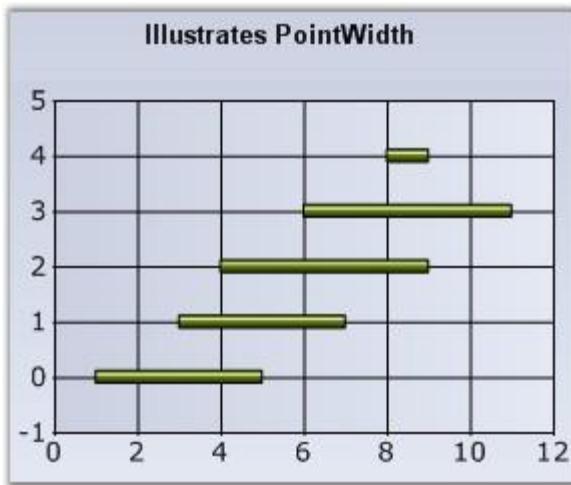


Figure 177: Chart with PointWidth= 0.25f

### Specific Data Point Setting

You can also set the PointWidth for specific points using **Series.Styles[0].PointWidth** for the first data point, **Series.Styles[1].PointWidth** for the second data point and so on.

#### [C#]

```
ganttSeries.Styles[0].PointWidth = 0.25f;  
ganttSeries.Styles[1].PointWidth = 0.5f;
```

#### [VB.NET]

```
Private ganttSeries.Styles(0).PointWidth = 0.25f  
Private ganttSeries.Styles(1).PointWidth = 0.5f
```

### See Also

[Gantt Chart](#)

### 4.5.1.56 PriceDownColor

Specifies a color for the financial item whose price is down.

Details	
<b>Possible Values</b>	Any valid color
<b>Default Value</b>	<b>Color.Red</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any series
<b>Applies to Chart Types</b>	Kagi Chart, Point and Figure Chart, Renko Chart, Three Line Break Chart

Here is code snippet using **PriceDownColor** in PointandFigure Chart.

**[C#]**

```
series7.PriceDownColor = Color.Magenta;  
series7.PriceUpColor= Color.Orange;
```

**[VB .NET]**

```
series7.PriceDownColor = Color.Magenta  
series7.PriceUpColor = Color.Orange
```



Figure 178: `PriceDownColor = "Magenta"`

#### See Also

[Kagi Chart](#), [Point and Figure Chart](#), [Three Line Break Chart](#), [Renko Chart](#)

#### 4.5.1.57 PriceUpColor

Specifies a color for the financial item whose price is up.

Details	
Possible Values	Any valid color
Default Value	<code>Color.Green</code>

<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any series
<b>Applies to Chart Types</b>	Kagi Chart, PointandFigure Chart, Renko Chart, Three Line Break Chart

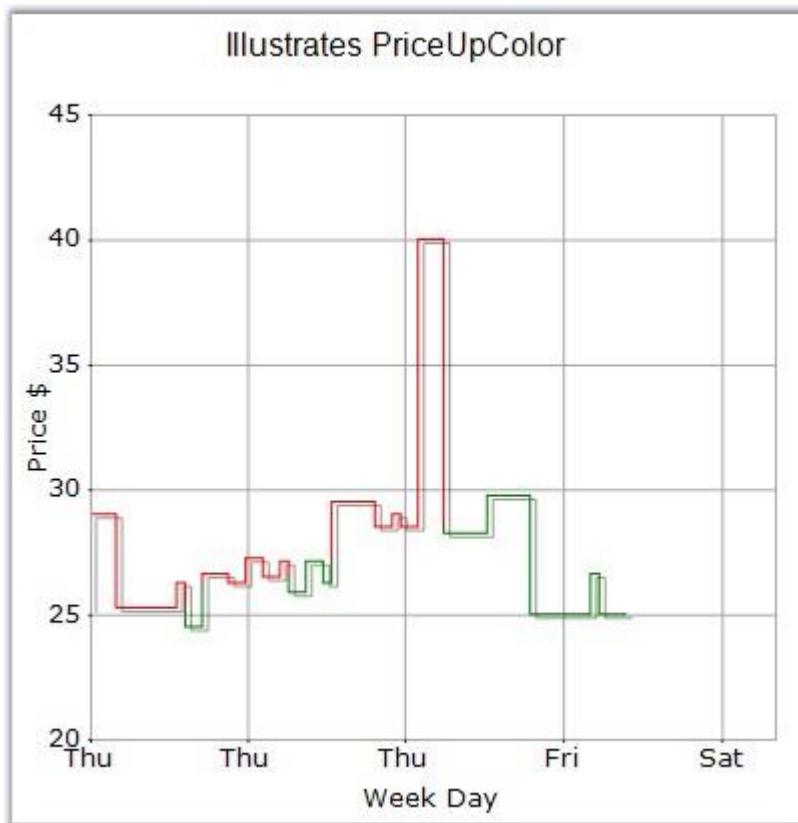
Here is sample code snippet using PriceUpColor in Kagi Chart.

**[C#]**

```
series.PriceUpColor = Color.Red;  
series.PriceDownColor = Color.Green;
```

**[VB .NET]**

```
series.PriceUpColor = Color.Red  
series.PriceDownColor = Color.Green
```



*Figure 179: PriceUpColor = "Red"*

**See Also**

[Kagi Chart](#), [Point and Figure Chart](#), [Three Line Break Chart](#), [Renko Chart](#)

#### 4.5.1.58 PyramidMode

Specifies the mode in which the y values should be interpreted. In Linear mode the y values are proportional to the length of the sides of the Pyramid. In Surface Mode the y values are proportional to the surface area of the corresponding blocks.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>Linear</b> - Draw pyramid chart in linear mode</li> <li>• <b>Surface</b> - Draw pyramid chart in surface mode</li> </ul>
<b>Default Value</b>	Linear
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Pyramid

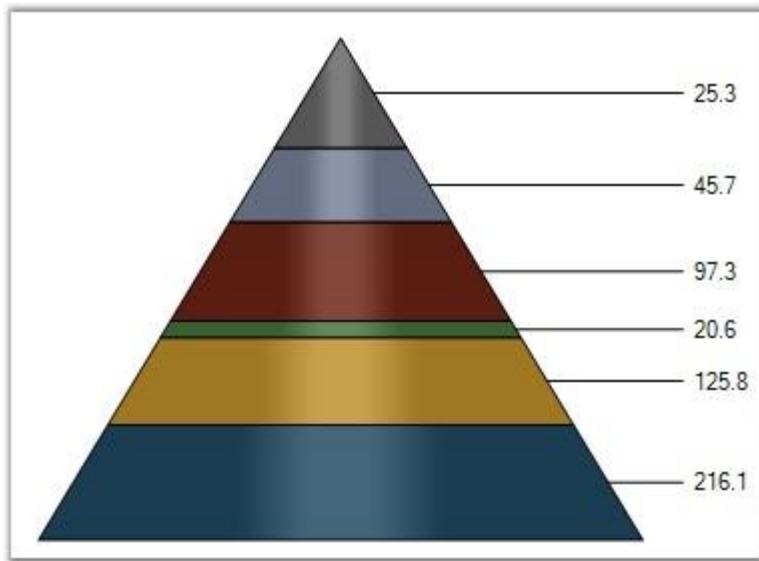
Here is some sample code.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.PyramidItem.PyramidMode=ChartPyramidMode.Surface;
```

**[VB .NET]**

```
Private
Me.chartControl1.Series(0).ConfigItems.PyramidItem.PyramidMode=ChartPyramidMode.Surface
```



*Figure 180: Surface Mode Pyramid Chart*

**See Also**

[Pyramid Chart](#)

#### 4.5.1.59 Radar Type

Indicates the type of radar chart to be rendered.

Details	
Possible Values	<ul style="list-style-type: none"><li><b>Area</b> - Renders the radar chart such that the points are connected and the enclosed region is not filled.</li><li><b>Line</b> - Renders the radar chart such that the points are connected but the enclosed region is not filled.</li><li><b>Symbol</b> - Points are rendered with the associated symbols</li></ul>
Default Value	Area

<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any series
<b>Applies to Chart Types</b>	Polar and Radar Chart

Here is code snippet using RadarType.

**[C#]**

```
this.chartControl1.Series[0].ConfigItems.RadarItem.Type =  
    ChartRadarDrawType.Symbol;  
this.chartControl1.Series[1].ConfigItems.RadarItem.Type =  
    ChartRadarDrawType.Symbol;  
this.chartControl1.Series[0].Style.Symbol.Shape =  
    ChartSymbolShape.Star;  
this.chartControl1.Series[1].Style.Symbol.Shape =  
    ChartSymbolShape.Star;  
this.chartControl1.Series[0].Style.Symbol.Color = Color.Blue;  
this.chartControl1.Series[1].Style.Symbol.Color = Color.Green;
```

**[VB .NET]**

```
Private Me.chartControl1.Series(0).ConfigItems.RadarItem.Type =  
    ChartRadarDrawType.Symbol  
Private Me.chartControl1.Series(1).ConfigItems.RadarItem.Type =  
    ChartRadarDrawType.Symbol  
Private Me.chartControl1.Series(0).Style.Symbol.Shape =  
    ChartSymbolShape.Star  
Private Me.chartControl1.Series(1).Style.Symbol.Shape =  
    ChartSymbolShape.Star  
Private Me.chartControl1.Series(0).Style.Symbol.Color = Color.Blue  
Private Me.chartControl1.Series(1).Style.Symbol.Color = Color.Green
```

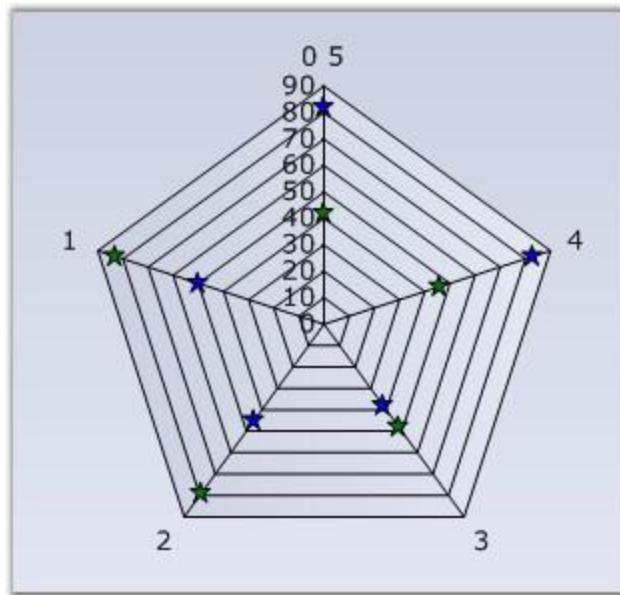


Figure 181: Radar Chart

**See Also**

[Polar and Radar Charts](#)

#### 4.5.1.60 RadarStyle

Indicates the style of the radar chart axes.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>Polygon</b> - Axes are renders as polygon.</li> <li>• <b>Circle</b> - Axes are renders as Circle.</li> </ul>
<b>Default Value</b>	<b>Polygon</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any series
<b>Applies to Chart Types</b>	Radar Chart

Here is code snippet using RadarType.

[C#]

```
this.chartControl1.RadarStyle = ChartRadarAxisStyle.Circle;
```

[VB .NET]

```
Me.chartControl1.RadarStyle = ChartRadarAxisStyle.Circle
```

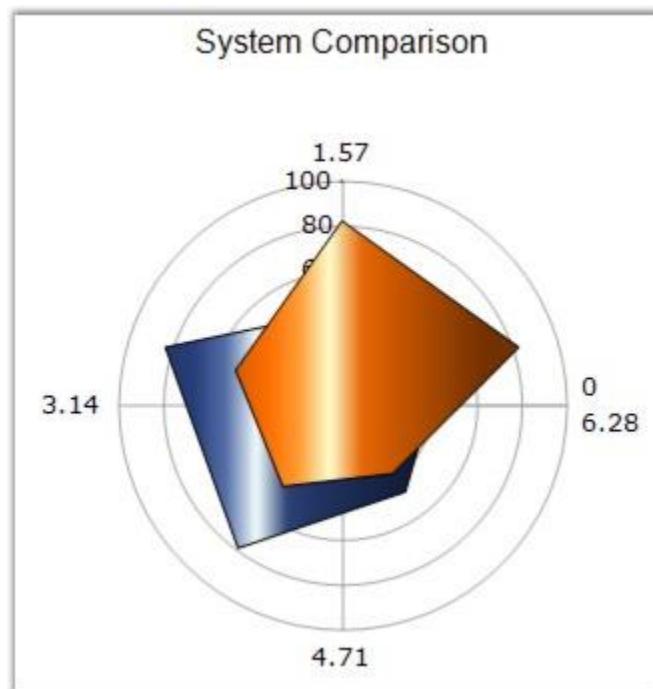
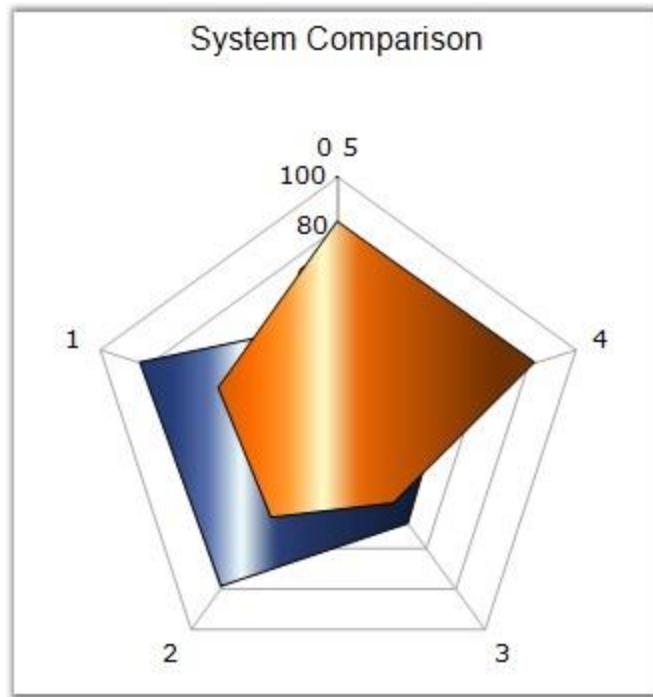


Figure 182: Radar Chart with Circle RadarStyle



*Figure 183: Radar Chart with Polygon RadarStyle*

#### See Also

[Radar Charts](#)

#### 4.5.1.61 RelatedPoints

Lets you specific the relationship between two points in the Gantt chart type. This will render a line connecting the specified points.

Details	
Possible Values	A ChartRelatedPointInfo object, which has the following properties: <ul style="list-style-type: none"><li>• <b>Color</b> - Any Color Object</li><li>• <b>Alignment</b> - Any PenAlignment Property</li><li>• <b>Points</b> - Integer Array containing the points which are to be connected.</li></ul>

	<ul style="list-style-type: none"> <li><b>Count</b> - Specifies the Number of points</li> <li><b>DashStyle</b> - Any System.Drawing.Drawing2D.DashStyle</li> <li><b>DashPattern</b> - A Float array with two values</li> <li><b>Width</b> - Any Float value</li> </ul>
<b>Default Value</b>	<ul style="list-style-type: none"> <li><b>Color</b> - Control Text Color</li> <li><b>Alignment</b> - Center</li> <li><b>Points</b> - Null</li> <li><b>Count</b> - 0</li> <li><b>DashStyle</b> - Solid</li> <li><b>DashPattern</b> - Null</li> <li><b>Width</b> - 5.0f</li> </ul>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series and Points
<b>Applies to Chart Types</b>	Gantt Chart

Here is sample code snippet using RelatedPoints.

[C#]

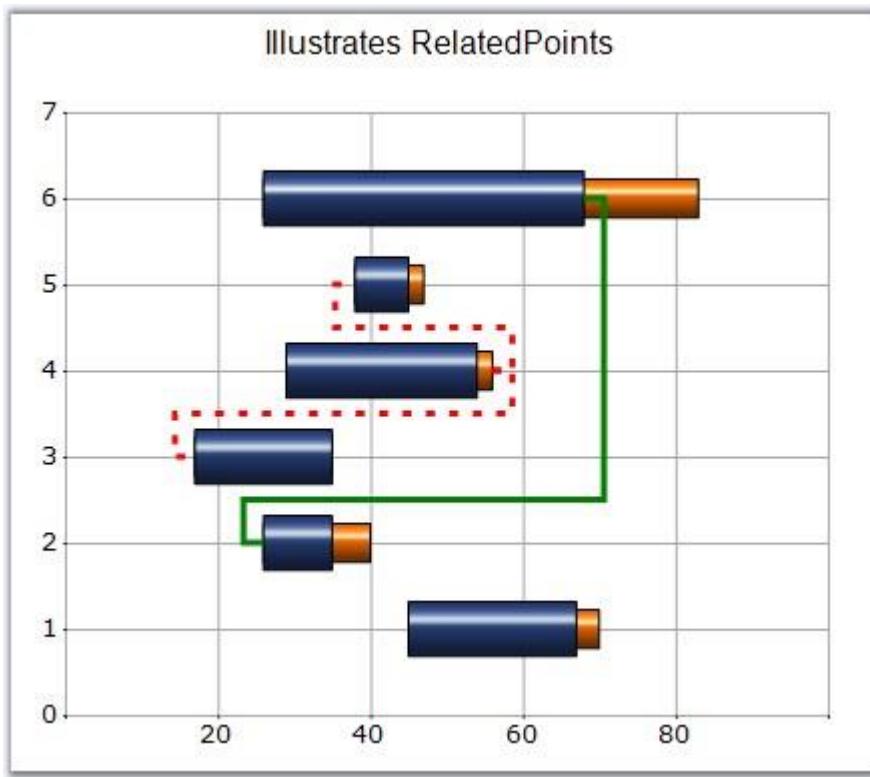
```
// Related Points for first series
int[] ptIndices = new int[] {2,4};
this.chartControl1.Series[0].Styles[3].RelatedPoints.Points =
ptIndices;
this.chartControl1.Series[0].Styles[3].RelatedPoints.Color = Color.Red;
this.chartControl1.Series[0].Styles[3].RelatedPoints.Alignment =
System.Drawing.Drawing2D.PenAlignment.Right;
this.chartControl1.Series[0].Styles[3].RelatedPoints.DashStyle =
System.Drawing.Drawing2D.DashStyle.Custom;
this.chartControl1.Series[0].Styles[3].RelatedPoints.Width = 3f;
float[] dash = new float[] { 1.5f, 2.4f };
this.chartControl1.Series[0].Styles[3].RelatedPoints.DashPattern =
dash;

// Related Points for second series
int[] ptIndices = new int[] { 1 };
this.chartControl1.Series[1].Styles[5].RelatedPoints.Points =
```

```
ptIndices;
this.chartControl1.Series[1].Styles[5].RelatedPoints.Color =
Color.Green;
this.chartControl1.Series[1].Styles[5].RelatedPoints.Alignment =
System.Drawing.Drawing2D.PenAlignment.Center;
this.chartControl1.Series[1].Styles[5].RelatedPoints.DashStyle =
System.Drawing.Drawing2D.DashStyle.Solid;
this.chartControl1.Series[1].Styles[5].RelatedPoints.Width = 3f;
```

**[VB.NET]**

```
' Related Points for first series
Dim ptIndices As Integer() = New Integer() {2,4}
Me.chartControl1.Series(0).Styles(3).RelatedPoints.Points = ptIndices
Me.chartControl1.Series(0).Styles(3).RelatedPoints.Color = Color.Red
Me.chartControl1.Series(0).Styles(3).RelatedPoints.Alignment =
System.Drawing.Drawing2D.PenAlignment.Right
Me.chartControl1.Series(0).Styles(3).RelatedPoints.DashStyle =
System.Drawing.Drawing2D.DashStyle.Custom
Me.chartControl1.Series(0).Styles(3).RelatedPoints.Width = 3f
Dim dash As Single() = New Single() { 1.5f, 2.4f }
Me.chartControl1.Series(0).Styles(3).RelatedPoints.DashPattern = dash
' Related Points for second series
Dim ptIndices As Integer() = New Integer() { 1 }
Me.chartControl1.Series(1).Styles(5).RelatedPoints.Points = ptIndices
Me.chartControl1.Series(1).Styles(5).RelatedPoints.Color = Color.Green
Me.chartControl1.Series(1).Styles(5).RelatedPoints.Alignment =
System.Drawing.Drawing2D.PenAlignment.Center
Me.chartControl1.Series(1).Styles(5).RelatedPoints.DashStyle =
System.Drawing.Drawing2D.DashStyle.Solid
Me.chartControl1.Series(1).Styles(5).RelatedPoints.Width = 3f
```



*Figure 184: Gantt Chart with RelatedPoints specified for certain Data Points*

#### See Also

[Gantt Chart](#)

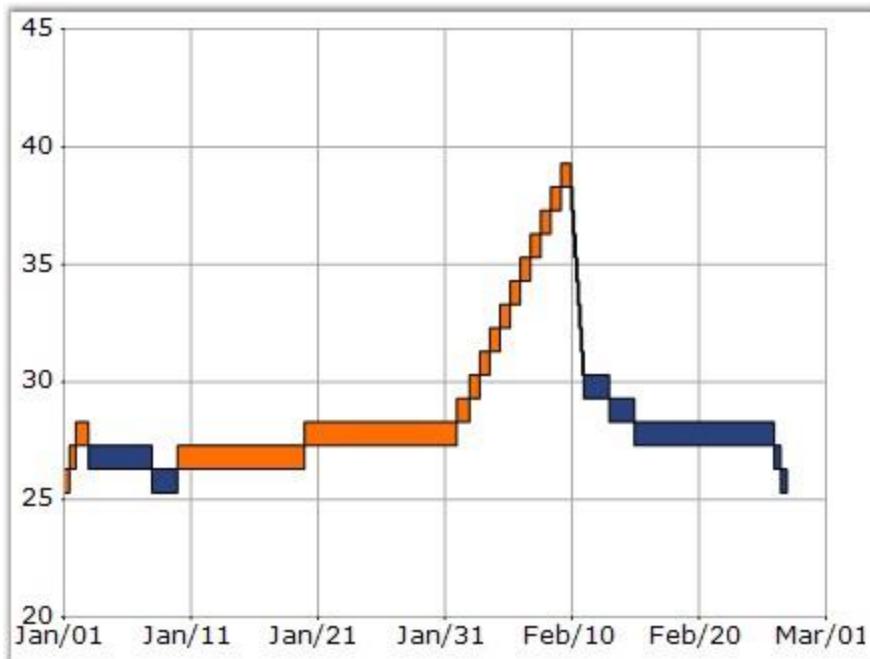
#### 4.5.1.62 ReversalAmount

Gets or sets the reversal amount for financial charts.

Details	
Possible Values	Any numeric value
Default Value	1
2D / 3D Limitations	No
Applies to Chart Element	Any Series

<b>Applies to Chart Types</b>	Kagi Chart, Three Line Break Chart, Point and Figure Chart, Renko Chart
-------------------------------	-------------------------------------------------------------------------

Here is code snippet using ReversalAmount in Renko Chart.



*Figure 185: Renko Chart with default ReversalAmount = "1"*

**[C#]**

```
series.ReversalAmount = 3;
```

**[VB .NET]**

```
Private series.ReversalAmount = 3
```

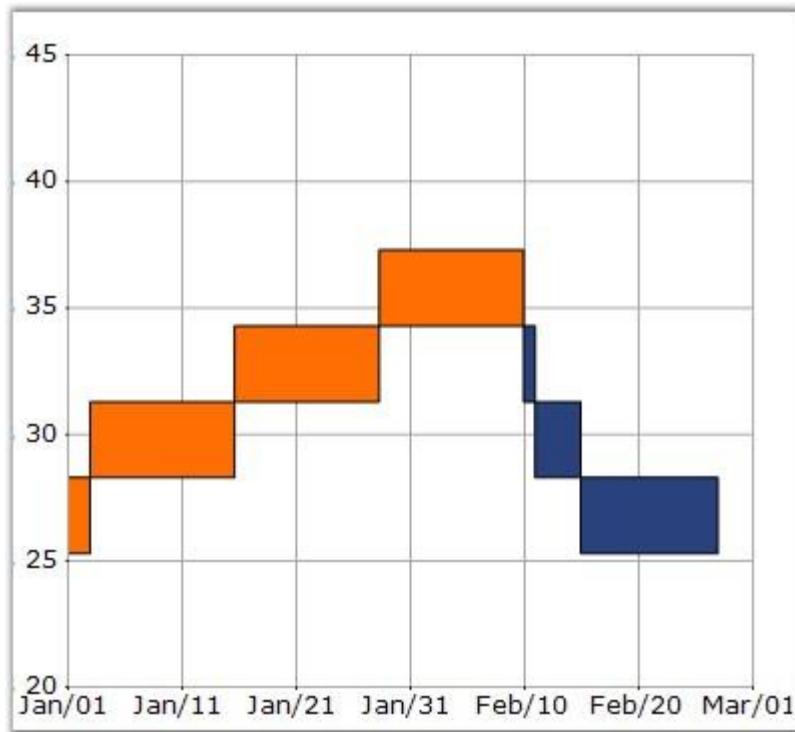


Figure 186: Renko Chart with *ReversalAmount* = "3"

#### See Also

[Kagi Chart](#), [Point and Figure Chart](#), [Three Line Break Chart](#), [Renko Chart](#)

#### 4.5.1.63 Rotate

Indicates whether the x and y axis should be rotated for this series.

Details	
Possible Values	True - Enable the Rotate property False - Disable the Rotate property
Default Value	False
2D / 3D Limitations	No

Applies to Chart Element	Any Series
Applies to Chart Types	Column Charts, Bar Charts, Area charts, Line Chart, Spline Chart, Stepline Chart, BoxandWhisker chart, Histogram chart, Polar and Radar Chart, Bubble And Scatter Chart

Here is sample code snippet using Rotate in Column Chart.

[C#]

```
this.chartControl1.Series[0].Rotate = true;
```

[VB .NET]

```
Private Me.chartControl1.Series(0).Rotate = True
```

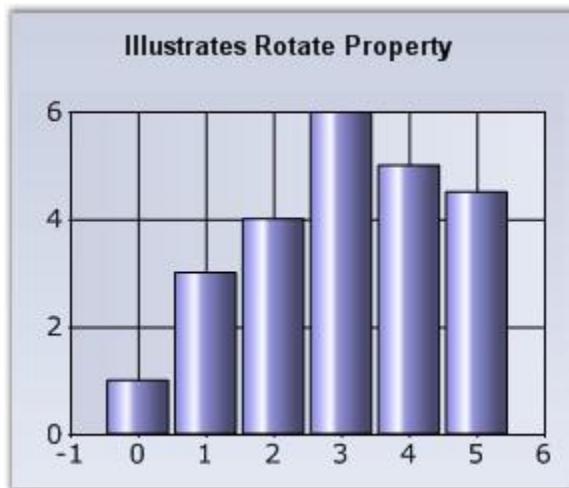
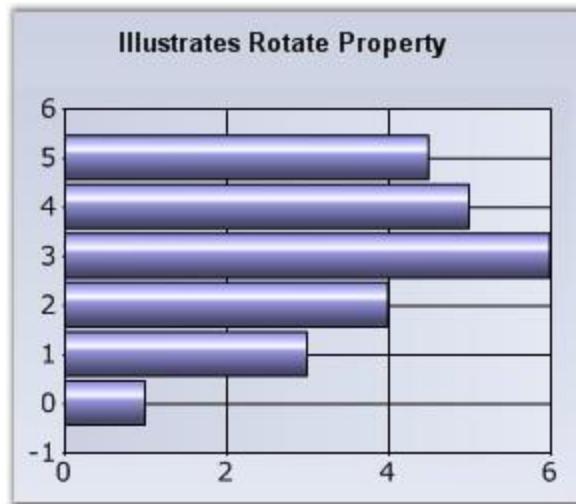


Figure 187: Column Chart



*Figure 188: Rotated Column Chart*

#### **See Also**

[Column Charts](#), [Bar Charts](#), [Area charts](#), [Line Chart](#), [Spline Chart](#), [Stepline Chart](#), [Kagi Chart](#), [BoxandWhisker chart](#), [Histogram chart](#), [Polar and Radar Chart](#), [Bubble And Scatter Chart](#)

#### **4.5.1.64 ScatterConnectType**

Specifies the connection type of the Scatter Charts.

Details	
<b>Possible Values</b>	None - Scatter Connect Type will be none. Line - Scatter Connect Type will be Line. Spline - Scatter Connect Type will be spline.
<b>Default Value</b>	<b>None</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	Scatter Chart

#### **Scatter Line Chart**

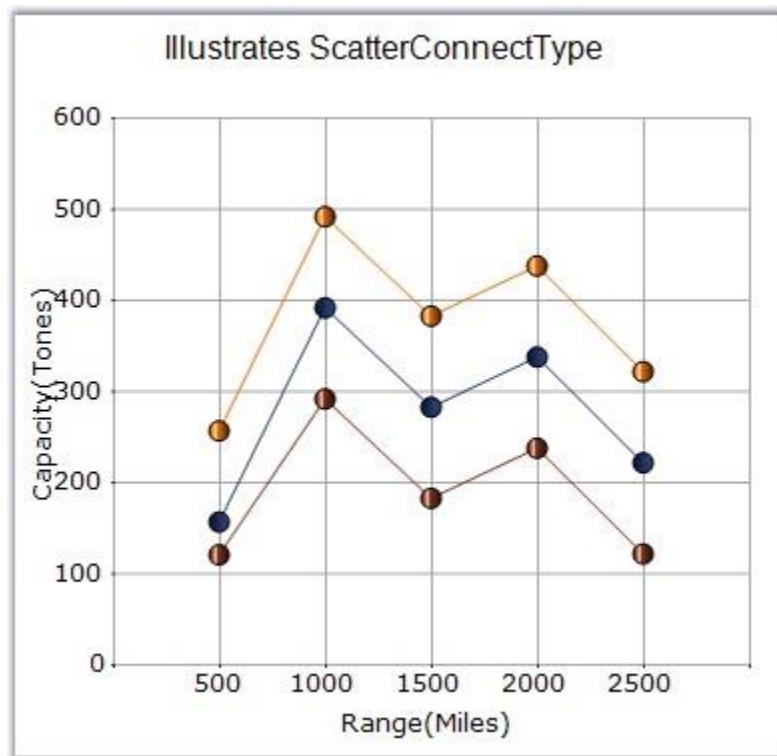
Optionally, you can connect the points in the series through straight lines using the **ScatterConnectType** property as shown below.

**[C#]**

```
series.ScatterConnectType = ScatterConnectType.Line;
```

**[VB .NET]**

```
series.ScatterConnectType = ScatterConnectType.Line
```



*Figure 189: Scatter Line Chart*

### Scatter Spline Chart

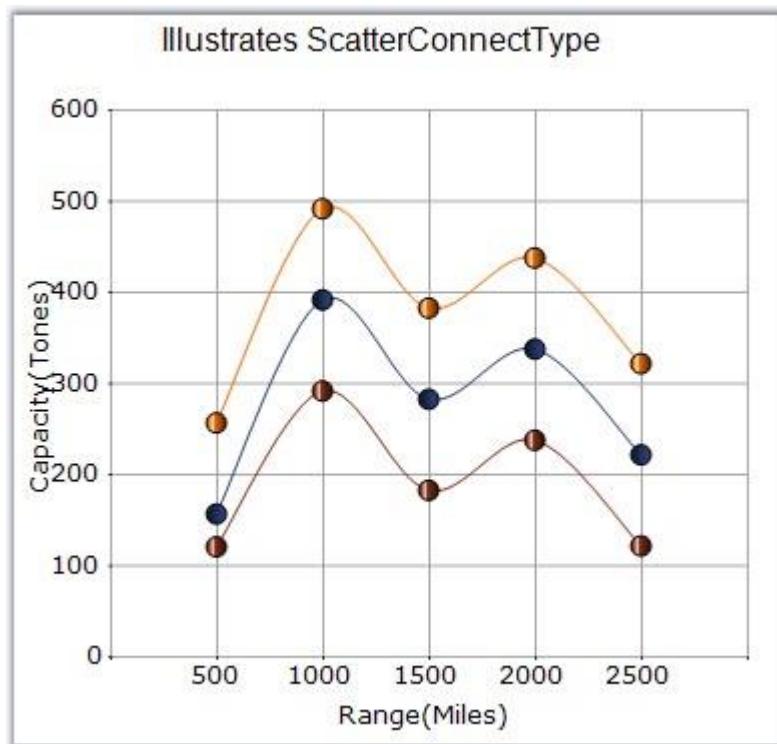
Alternatively, you can connect the points in the series through splines using the **ScatterConnectType** property as shown below.

**[C#]**

```
series.ScatterConnectType = ScatterConnectType.Spline;  
series.ScatterSplineTension = 1; // Default is 0
```

**[VB .NET]**

```
series.ScatterConnectType = ScatterConnectType.Spline  
series.ScatterSplineTension = 1 ' Default is 0
```



*Figure 190: Scatter Spline Chart*

**See Also**

[Scatter Chart](#)

#### 4.5.1.65 ScatterSplineTension

Sets the tension required for the Scatter Spline Chart.

Details	
<b>Possible Values</b>	Any Possible Numeric Values
<b>Default Value</b>	0.5
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	ScatterSplineChart

Here is some sample code.

**[C#]**

```
this.chartControl1.Series[i].ScatterConnectType =  
ScatterConnectType.Spline;  
this.chartControl1.Series[i].ScatterSplineTension =3;
```

**[VB.NET]**

```
Private Me.chartControl1.Series(i).ScatterConnectType =  
ScatterConnectType.Spline  
Private Me.chartControl1.Series(i).ScatterSplineTension =3
```

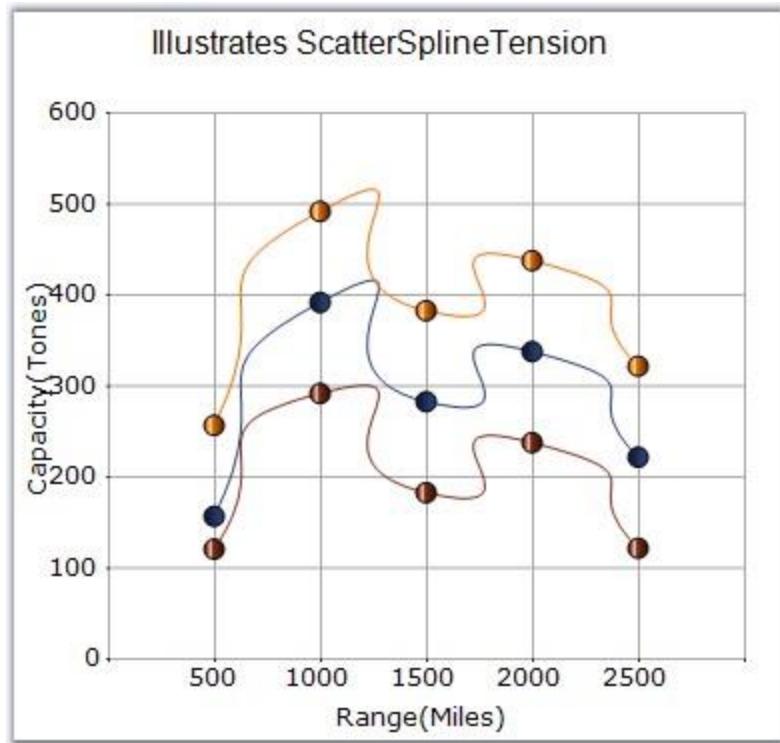


Figure 191: Scatter Chart with ScatterSplineTension = "3"

#### See Also

[Scatter Chart](#)

#### 4.5.1.66 SeriesToolTipFormat

Specifies the format for tooltip display in series.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"><li>{0} - Series Name</li><li>{1} - Series Style tooltip</li></ul>
<b>Default Value</b>	{0}
<b>2D / 3D Limitations</b>	No

<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Area Charts, Radar Chart, Polar Chart, ThreeLineBreak Chart, PointAndFigure Chart, StepLine Chart, Spline Chart, HiloOpenClose(3D), RotatedSpline, Kagi Chart

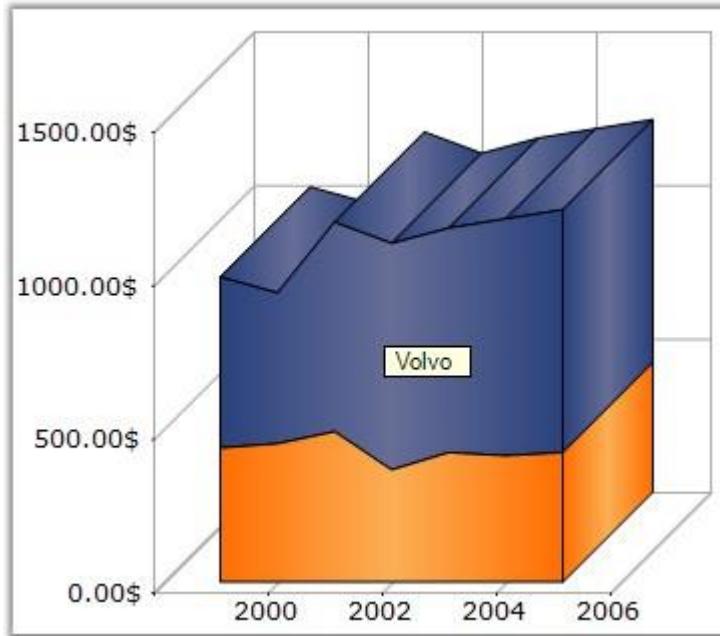
Here is some sample code.

**[C#]**

```
this.chartControl1.Series[1].SeriesToolTipFormat = "{0}";
```

**[VB .NET]**

```
Private Me.chartControl1.Series(1).SeriesToolTipFormat = "{0}"
```



*Figure 192: SeriesToolTipFormat set = "{0}"*

#### See Also

[Area Charts](#), [Radar Chart](#), [Polar Chart](#), [ThreeLineBreak Chart](#), [PointAndFigure Chart](#), [StepLine Chart](#), [Spline Chart](#), [HiloOpenClose\(3D\)](#), [RotatedSpline](#), [Kagi Chart](#)

#### 4.5.1.67 ShadingMode

Specifies the appearance of the chart series.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>FlatRectangle</b> - Displays in a flat rectangular format.</li> <li>• <b>PhongCylinder</b> - Displays in a cylindrical format.</li> </ul>
<b>Default Value</b>	<b>PhongCylinder</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Column Chart, BarCharts, Candle Chart, HiLO Chart, HiLoOpenClose Chart, Tornado chart, BoxandWhisker chart, Gantt Chart, Histogram Chart, Polar and Radar Chart

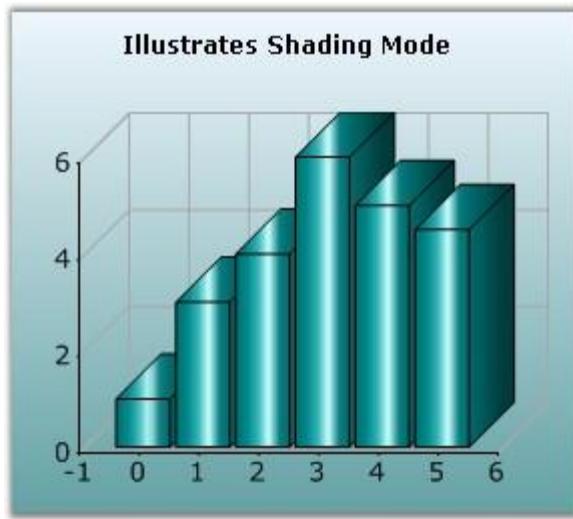
Here is sample code snippet using ShadingMode.

[C#]

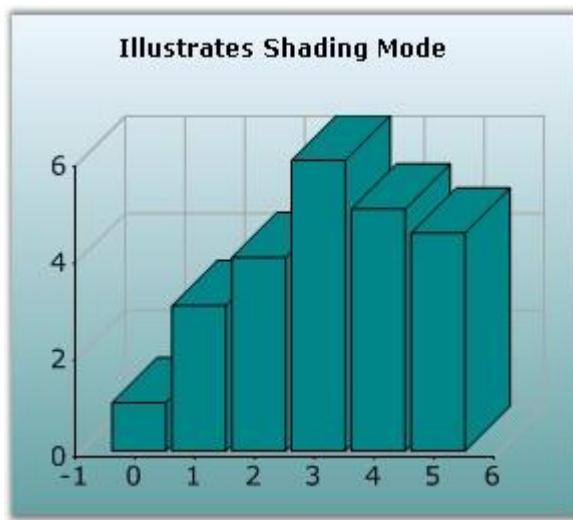
```
this.chartControl1.Series[0].ConfigItems.ColumnItem.ShadingMode =
ChartColumnShadingMode.FlatRectangle;
```

[VB .NET]

```
Private Me.chartControl1.Series(0).ConfigItems.ColumnItem.ShadingMode =
ChartColumnShadingMode.FlatRectangle
```



*Figure 193: Normal Column Chart (3D View)*



*Figure 194: Column Chart with ShadingMode = "FlatRectangle"*

#### See Also

[Column Charts](#), [BarCharts](#), [Candle Chart](#), [HiLO Chart](#), [HiLoOpenClose Chart](#), [Tornado chart](#), [BoxandWhisker chart](#), [Gantt Chart](#), [Histogram Chart](#), [Polar and Radar Chart](#)

#### 4.5.1.68 ShadowInterior

Specifies the interior color of the shadow.

Details	
<b>Possible Values</b>	Any valid Color format
<b>Default Value</b>	<b>Black</b>
<b>2D / 3D Limitations</b>	2D only
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Column Charts, Bubble Chart, Line Charts, BarCharts, Candle Chart, Kagi Chart, Point and Figure Chart, Renko Chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Pie Chart, Polar and Radar Chart, Area Chart, Step Area Chart

Here is sample code snippet using **ShadowInterior** in Column Chart.

### Series Wide Setting

[C#]

```
// Specifying Shadow Interior for 2 series
this.chartControl1.Series[0].Style.DisplayShadow = true;
this.chartControl1.Series[0].Style.ShadowInterior = new
BrushInfo(GradientStyle.None, Color.SteelBlue,Color.SteelBlue);
```

[VB .NET]

```
' Specifying Shadow Interior for 2 series
Private Me.chartControl1.Series(0).Style.DisplayShadow = True
Private Me.chartControl1.Series(0).Style.ShadowInterior = New
BrushInfo(GradientStyle.None, Color.SteelBlue,Color.SteelBlue)
```

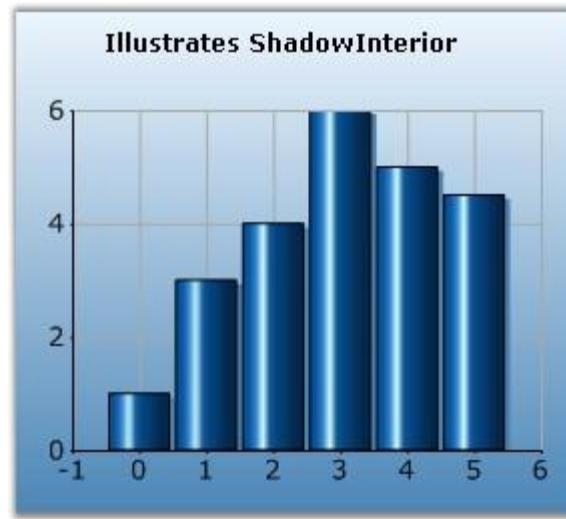


Figure 195: ColumnChart with SkyBlue Shadow

### Specific Data Point Setting

To specify different shadow colors for individual points, use **Series.Styles[0].ShadowInterior** property.

#### [C#]

```
this.chartControl1.Series[0].Styles[0].ShadowInterior = new  
BrushInfo(GradientStyle.None, Color.SteelBlue,Color.SteelBlue);  
this.chartControl1.Series[0].Styles[0].ShadowInterior = new  
BrushInfo(GradientStyle.None, Color.Gray,Color.Gray);
```

#### [VB .NET]

```
Private Me.chartControl1.Series(0).Style.ShadowInterior = New  
BrushInfo(GradientStyle.None, Color.SteelBlue,Color.SteelBlue)  
Private Me.chartControl1.Series(0).Style.ShadowInterior = New  
BrushInfo(GradientStyle.None, Color.Gray,Color.Gray)
```

### See Also

[Column Charts](#), [Bar Charts](#), [Line Charts](#), [Candle Chart](#), [Kagi Chart](#), [BoxandWhisker chart](#), [Histogram chart](#), [Polar and Radar Chart](#), [Point and Figure Chart](#), [Renko Chart](#), [Three Line Break Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Pie Chart](#), [Area Chart](#), [Step Area Chart](#)

#### 4.5.1.69 ShadowOffset

Specifies the width of the shadow of the series.

Details	
<b>Possible Values</b>	Any possible numeric values for x, y
<b>Default Value</b>	3, 2
<b>2D / 3D Limitations</b>	2D only
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Column Charts, Bubble Chart, Line Charts, BarCharts, Candle Chart, Kagi Chart, Point and Figure Chart, Renko Chart, Three Line Break Chart, Box and Whisker Chart, Gantt Chart, Histogram Chart, Tornado Chart, Pie Chart, Polar and Radar Chart, Area Chart, Step Area Chart

Here is sample code snippet using **ShadowOffset** in Column Chart.

##### Series Wide Setting

```
[C#]

series.Style.DisplayShadow = true;
series.Style.ShadowOffset = new Size(7, 7);

//For specific points
series.Styles[0].ShadowOffset = new Size(7, 7);
series.Styles[1].ShadowOffset = new Size(8, 8);
series.Styles[2].ShadowOffset = new Size(6, 6);
```

```
[VB .NET]

Private series.Style.DisplayShadow = True
Private series.Style.ShadowOffset = New Size(7, 7)

'For specific points
Private series.Styles(0).ShadowOffset = New Size(7, 7)
```

```
Private series.Styles(1).ShadowOffset = New Size(8, 8)
Private series.Styles(2).ShadowOffset = New Size(6, 6)
```

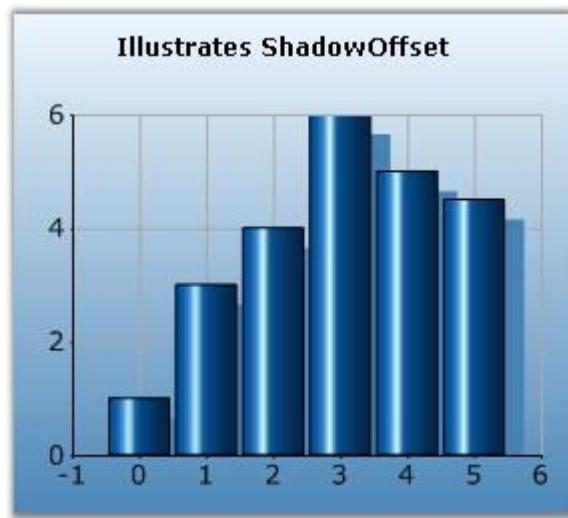


Figure 196: ColumnChart with ShadowOffset (7,7)

### Specific Data Point Setting

[C#]

```
//For specific points
series.Styles[0].ShadowOffset = new Size(7, 7);
series.Styles[1].ShadowOffset = new Size(8, 8);
series.Styles[2].ShadowOffset = new Size(6, 6);
```

[VB.NET]

```
'For specific points
Private series.Styles(0).ShadowOffset = New Size(7, 7)
Private series.Styles(1).ShadowOffset = New Size(8, 8)
Private series.Styles(2).ShadowOffset = New Size(6, 6)
```

### See Also

[Column Charts](#), [Bar Charts](#), [Line Charts](#), [Candle Chart](#), [Kagi Chart](#), [BoxandWhisker chart](#), [Histogram chart](#), [Polar and Radar Chart](#), [Point and Figure Chart](#), [Renko Chart](#), [Three Line Break Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Pie Chart](#), [Area Chart](#), [Step Area Chart](#)

#### 4.5.1.70 ShowDataBindLabels

Indicates whether data bound labels are displayed in the chart.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• True - Displays the databind labels.</li> <li>• False - Hides the databind labels.</li> </ul>
<b>Default Value</b>	False
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All Series
<b>Applies to Chart Types</b>	Pie Chart, Doughnut Chart, Funnel Chart and Pyramid chart.

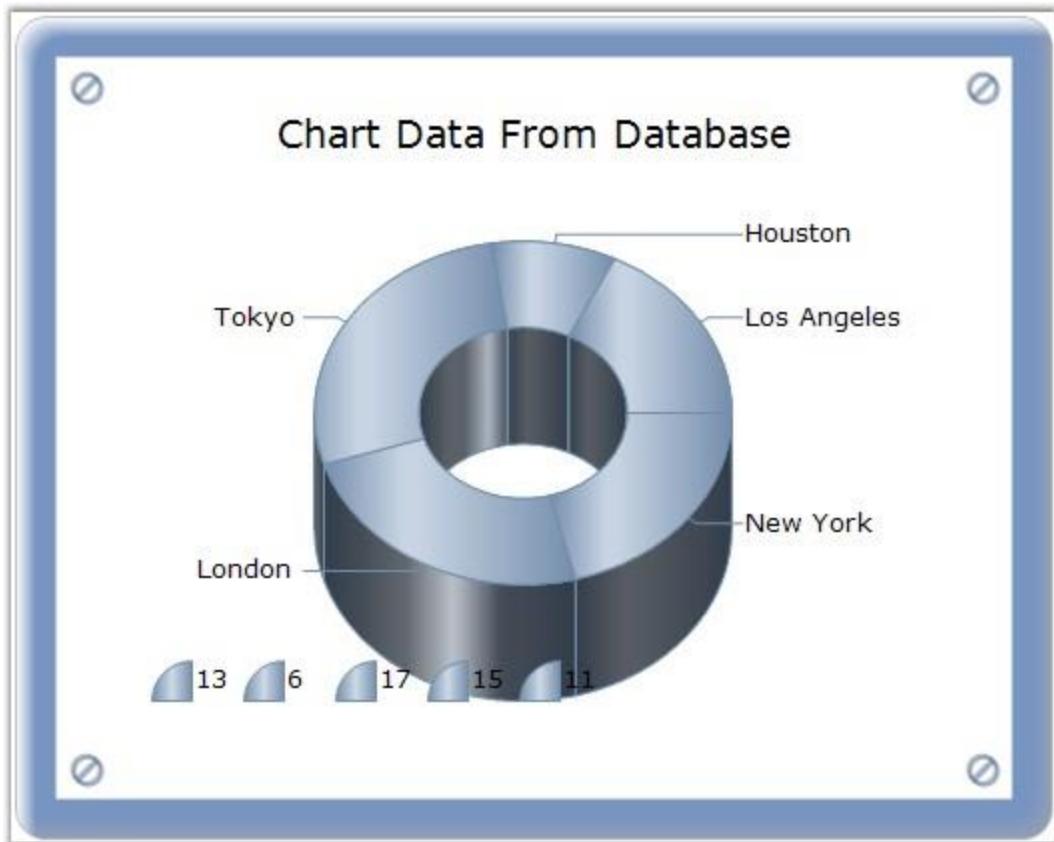
Here is sample code snippet using ShowDataPointLabels.

##### [C#]

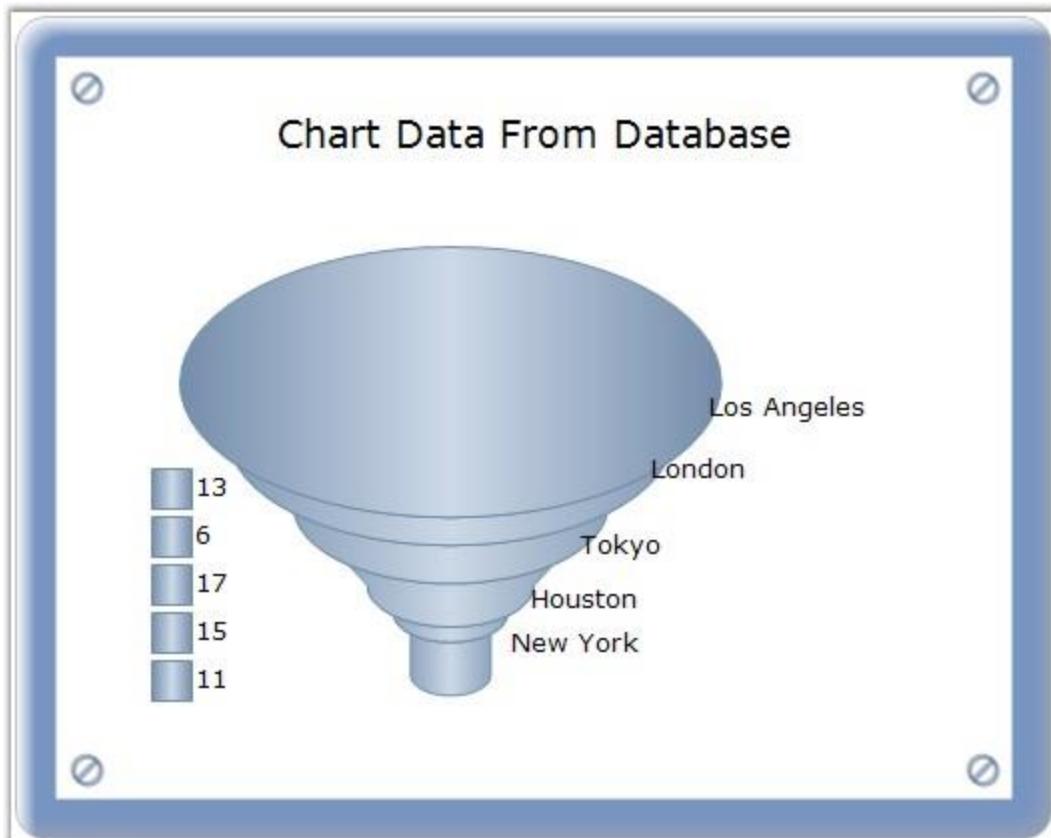
```
//For Pie Chart
this.chartControl.Series[0].ConfigItems.PieItem.ShowDataBindLabels =
true;
//For Funnel Chart
this.chartControl.Series[0].ConfigItems.FunnelItem.ShowDataBindLabels =
true;
//For Pyramid Chart
this.chartControl.Series[0].ConfigItems.PyramidItem.ShowDataBindLabels =
true;
```

##### [VB .NET]

```
'For Pie Chart
Me.chartControl.Series(0).ConfigItems.PieItem.ShowDataBindLabels = True
'For Funnel Chart
Me.chartControl.Series(0).ConfigItems.FunnelItem.ShowDataBindLabels =
True
'For Pyramid Chart
Me.chartControl.Series(0).ConfigItems.PyramidItem.ShowDataBindLabels =
True
```



*Figure 197: Doughnut Chart with Data-Bound Labels*



*Figure 198: Funnel Chart with Data-Bound Labels*

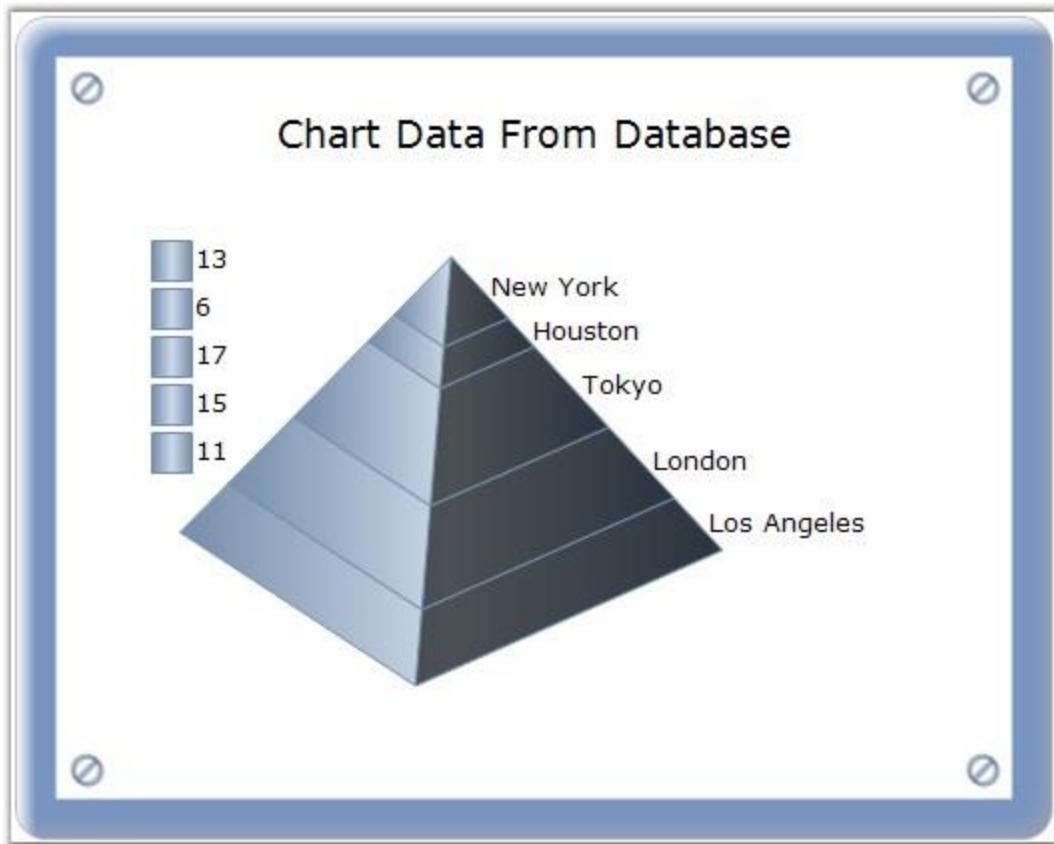


Figure 199: Pyramid Chart with Data-Bound Labels

#### See Also

[Pie Chart](#), [Doughnut Chart](#), [Funnel Chart](#), [Pyramid Chart](#)

#### 4.5.1.71 ShowHistogramDataPoints

Indicates if the histogram data points should be shown.

Details	
Possible Values	<ul style="list-style-type: none"><li>• <b>True</b> - Displays the datapoints.</li><li>• <b>False</b> - Hides the datapoints.</li></ul>
Default Value	True

<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Histogram Chart

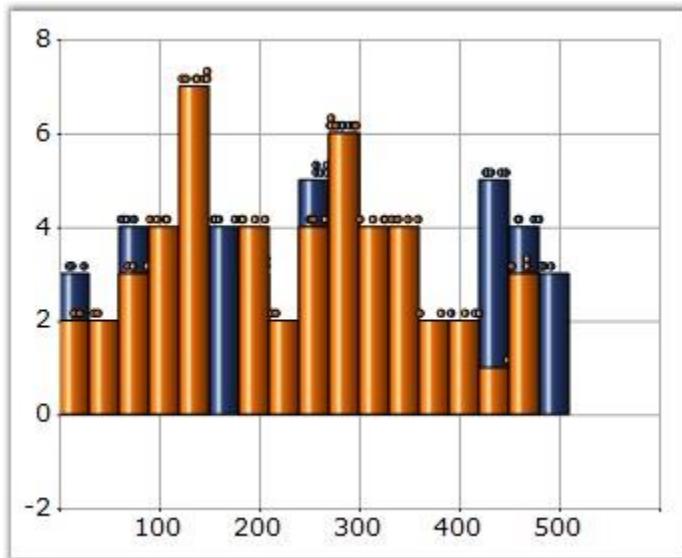
Here is sample code snippet using `ShowHistogramDataPoints`.

**[C#]**

```
this.chartControl1.Series[0].ShowHistogramDataPoints =true;
```

**[VB .NET]**

```
Private Me.chartControl1.Series(0).ShowHistogramDataPoints =True
```



*Figure 200: ShowHistogramDataPoints set to True*

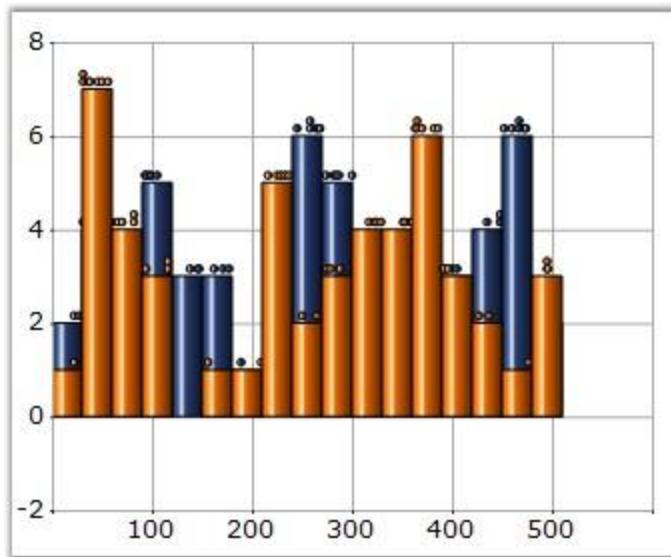


Figure 201: ShowHistogramDataPoints set to False

## See Also

[Histogram Chart](#)

### 4.5.1.72 ShowTicks

Indicates whether ticks should be shown or not.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>True</b> - Displays ticks</li> <li>• <b>False</b> - Hides ticks</li> </ul>
<b>Default Value</b>	True
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Pie Chart

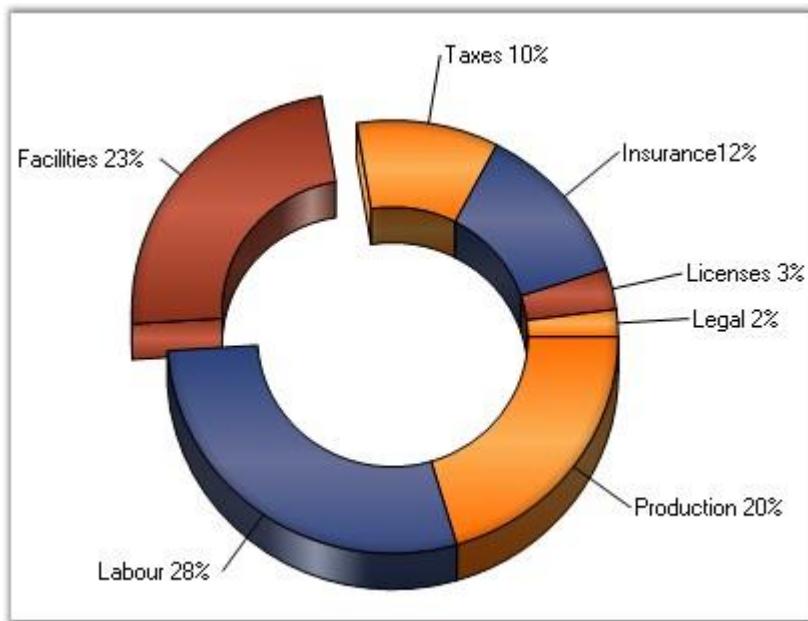
Here is a sample code snippet using ShowTicks.

**[C#]**

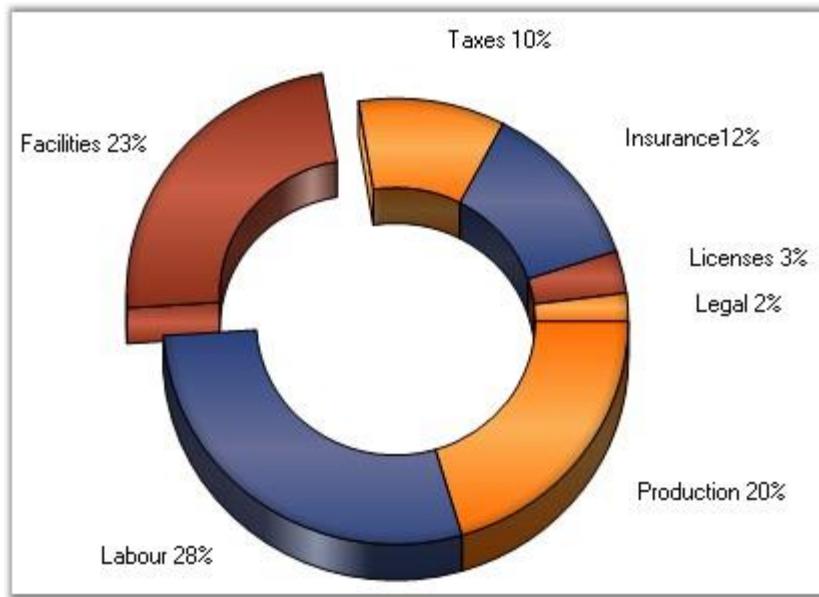
```
// Enables Ticks  
this.chartControl1.Series[0].ShowTicks = true;
```

**[VB .NET]**

```
' Enables Ticks  
Private Me.chartControl1.Series(0).ShowTicks = True
```



*Figure 202: ShowTicks = "True"*



*Figure 203: ShowTicks = "False"*

#### **See Also**

[Pie Chart](#)

#### **4.5.1.73 SmartLabels**

Specifies the behavior of the labels. If set to true, the labels will be rendered to avoid overlap with other labels.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"><li>• <b>True</b> - Enables smart labels</li><li>• <b>False</b> - Disables smart labels</li></ul>
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All chart types

Here is sample code snippet using Smart Labels in ColumnChart.

**[C#]**

```
this.chartControl1.Series[0].Style.DisplayText = true;
series.Styles[0].Text = series.Name;
this.chartControl1.Series[0].SmartLabels = true;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).Style.DisplayText = True
series.Styles(0).Text = series.Name
Private Me.chartControl1.Series(0).SmartLabels = True
```

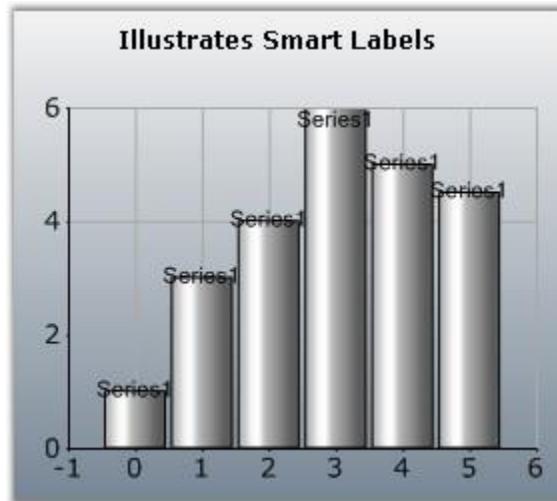


Figure 204: Chart without enabling SmartLabels

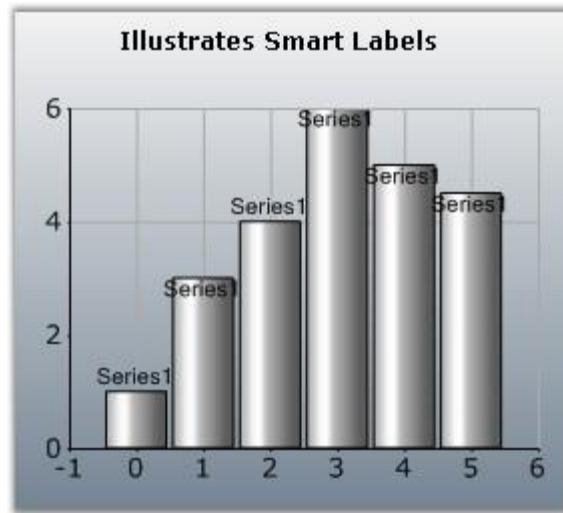


Figure 205: Smart Label Enabled Column Chart

### Custom borders for smart Labels

Smart labels can be made more smarter by displaying with customized borders. The color and the width of the border can be changed using the appearance properties available.

**SmartLabelsBorderColor** property is used to set color for the border and

**SmartLabelsBorderWidth** property is used to set the width of the border.

#### [C#]

```
this.chartControl1.Series[0].SmartLabelsBorderColor = Color.Yellow;  
this.chartControl1.Series[0].SmartLabelsBorderWidth = 2
```

#### [VB .NET]

```
Me.chartControl1.Series(0).SmartLabelsBorderColor = Color.Yellow  
Me.chartControl1.Series(0).SmartLabelsBorderWidth = 2
```

### See Also

[Chart Types](#)

### 4.5.1.74 Spacing

#### Spacing between data points

This specifies the space/width between data points in the x axis. This value is specified in percentage (%) of interval width. So, for example, if the value of the property is 20%, then only 80% of the interval width is used for rendering the data point(s). If there are multiple series then the available width is divided between the data points in the different series. This of course is used only for appropriate chart types like the column chart which has a width component.

This property will not be used when **ChartColumnWidth** is set to **FixedSizeMode**.

Details	
<b>Possible Values</b>	A double value (10 to 99)
<b>Default Value</b>	<b>30</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series and points
<b>Applies to Chart Types</b>	Column Charts, BarCharts, Box and Whisker Chart, Gantt Chart, Tornado Chart, Candle Chart, Hilo Chart, Hilo Open Close Chart

#### [C#]

```
//Indicates the spacing width in percentage that is to be applied
//between the datapoints of the column chart.
this.chartControl1.Spacing = 50;
```

#### [VB .NET]

```
'Indicates the spacing width in percentage that is to be applied
//between the data points of the column chart.
Me.chartControl1.Spacing = 50
```

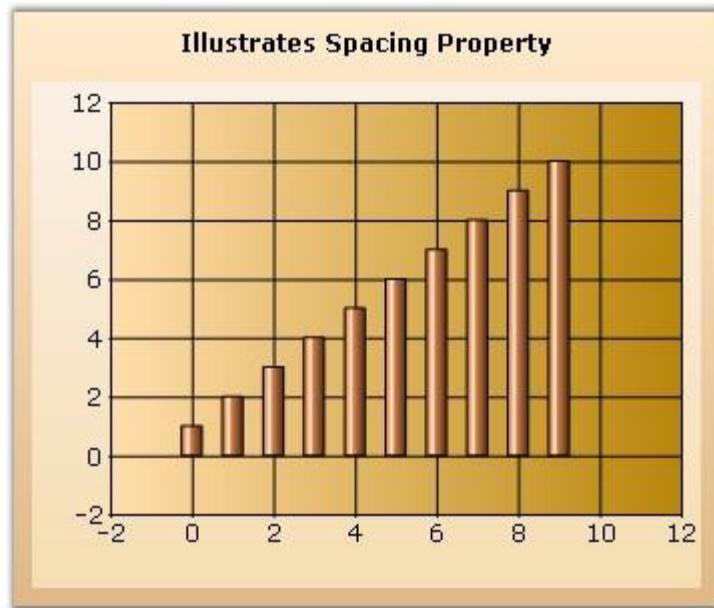


Figure 206: Series rendered with 50 percent Spacing

#### See Also

[Column Charts](#), [BarCharts](#), [Box and Whisker Chart](#), [Gantt Chart](#), [Tornado Chart](#), [Candle Chart](#), [Hilo Chart](#), [Hilo Open Close Chart](#)

#### 4.5.1.75 SpacingBetweenSeries

Essential Chart provides support to control the spacing between series using **SpacingBetweenSeries** property.

Details	
<b>Possible Values</b>	A double value (10 to 100)
<b>Default Value</b>	10
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Area Charts, BarCharts, Line Charts, Bubble Chart, Financial Charts, Gantt Chart, Histogram chart, Tornado Chart, Combination Chart, Box

and Whisker Chart

[C#]

```
//Specifies the spacing between individual series.  
this.chartControll1.SpacingBetweenSeries = 20;
```

[VB.NET]

```
'Specifies the spacing between individual series.  
Me.chartControll1.SpacingBetweenSeries = 20
```

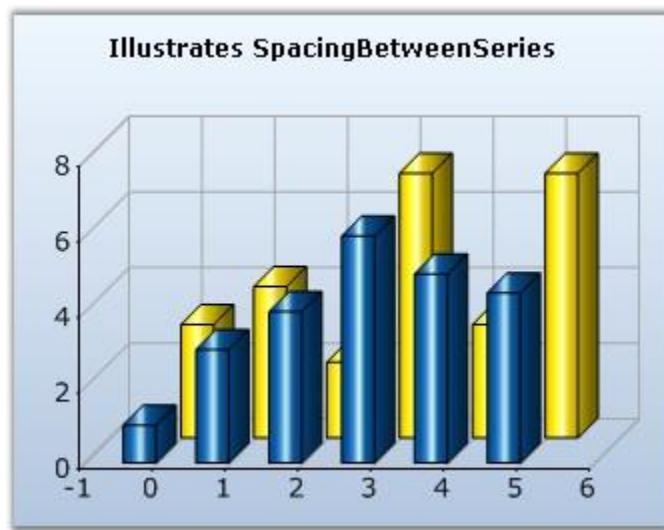


Figure 207: Chart with default Spacing between Series

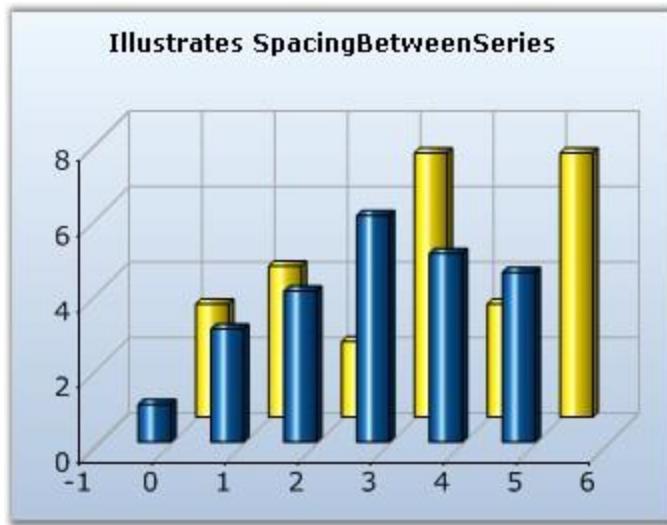


Figure 208: Chart with SpacingBetweenSeries = "80"

### See Also

[Area Charts](#), [BarCharts](#), [Line Charts](#), [Column Charts](#), [Bubble Chart](#), [Financial Charts](#), [Gantt Chart](#), [Histogram chart](#), [Tornado Chart](#), [Combination Chart](#), [Box and Whisker Chart](#)

### 4.5.1.76 SpacingBetweenPoints

Essential Chart provides support to control the spacing between points using **SpacingBetweenPoints** property.

Details	
<b>Possible Values</b>	A double value (0 to 99)
<b>Default Value</b>	<b>0</b>
<b>2D / 3D Limitations</b>	2D only
<b>Applies to Chart Element</b>	Series points
<b>Applies to Chart Types</b>	Column Chart, Bar Chart, HiLo Chart, HiLo Open Close Chart, Candle Chart, Tornado Chart, Boxes and Whisker Chart

[C#]

```
this.chartControl1.SpacingBetweenPoints = 70;
```

[VB.NET]

```
Me.chartControl1.SpacingBetweenSeries = 70
```

**See Also**

[Column Chart](#), [Bar Chart](#), [HiLo Chart](#), [HiLo Open Close Chart](#), [Candle Chart](#), [Tornado Chart](#), [Box and Whisker Chart](#)

#### 4.5.1.77 Stacking Group

This section illustrates how to group the stacking series with another stacking series.

1. In order to group the stacking series with another stacking series in chart control, you need to set a **StackingGroup** property of the chart series with the desired group name.

The below example demonstrates the code on setting the **StackingGroup** for the series in the Chart control.

[C#]

```
ChartSeries ser1 = new ChartSeries("Series 1");
ser1.Type = ChartSeriesType.StackingColumn;
// specifying group name .
ser1.StackingGroup = "FirstGroup";
ChartSeries ser2 = new ChartSeries("Series 2");
ser2.Type = ChartSeriesType.StackingColumn;
// specifying group name .
ser2.StackingGroup = "SecondGroup";
ChartSeries ser3 = new ChartSeries("Series 3");
ser3.Type = ChartSeriesType.StackingColumn;
// specifying group name .
ser3.StackingGroup = "FirstGroup";
```

[VB]

```
Dim ser1 As New ChartSeries("Series 1")
ser1.Type = ChartSeriesType.StackingColumn
' specifying group name .
ser1.StackingGroup = "FirstGroup"
Dim ser2 As New ChartSeries("Series 2")
ser2.Type = ChartSeriesType.StackingColumn
' specifying group name .
ser2.StackingGroup = "SecondGroup"
Dim ser3 As New ChartSeries("Series 3")
ser3.Type = ChartSeriesType.StackingColumn
' specifying group name .
ser3.StackingGroup = "FirstGroup"
```

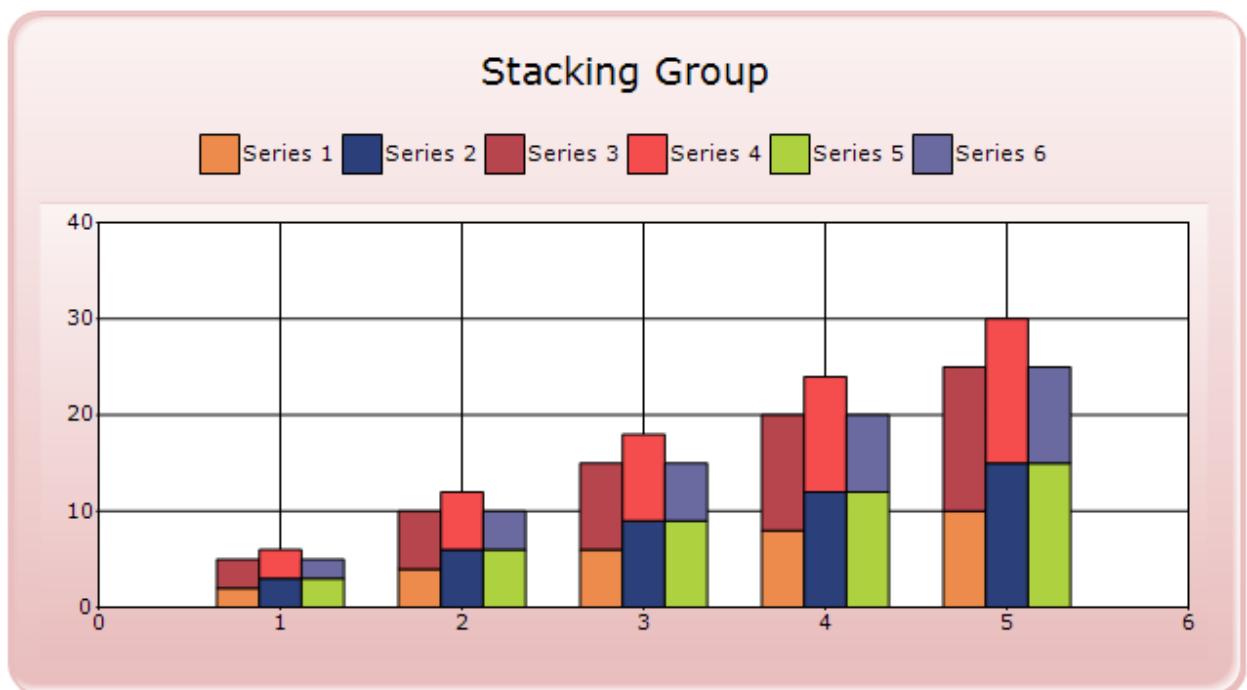


Figure 209: Column chart with stacking group.

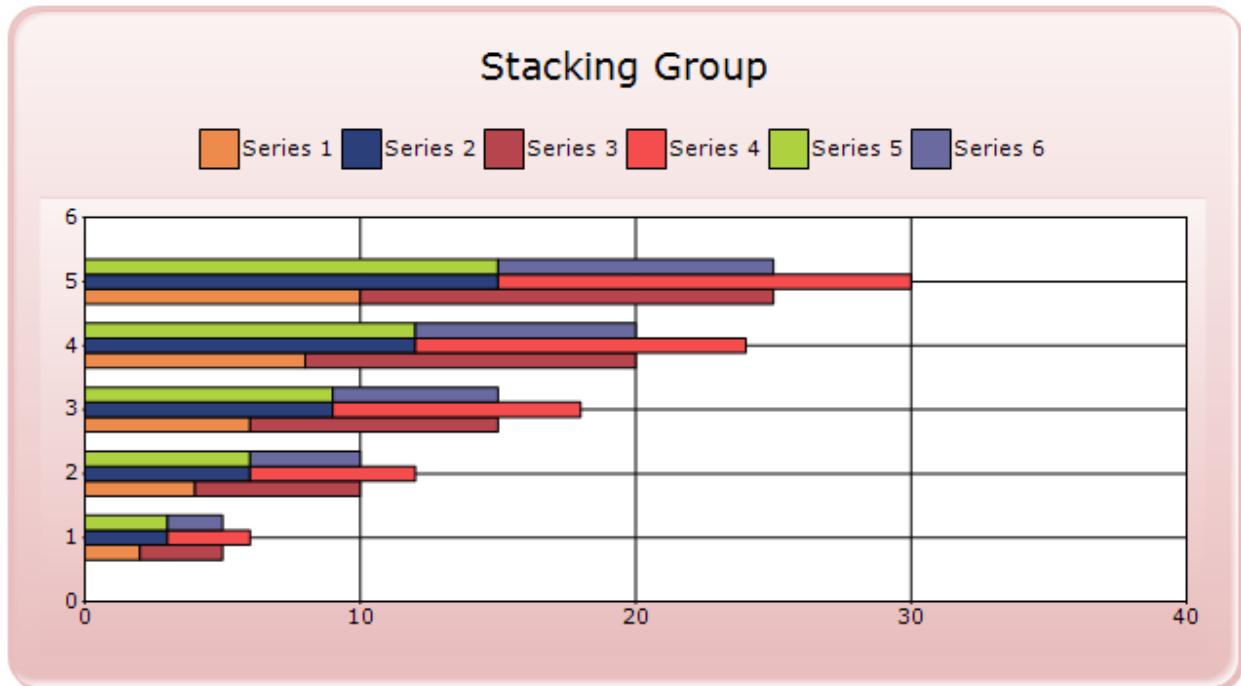


Figure 210: Bar chart with Stacking group.

#### 4.5.1.78 StepItem.Inverted

Specifies if the particular stepline is inverted or not in the StepAreaChart.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>True</b> - Indicates that the stepline is inverted.</li> <li>• <b>False</b> - Indicates that the stepline is not inverted.</li> </ul>
<b>Default Value</b>	True
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	StepAreaChart, StepLine Chart

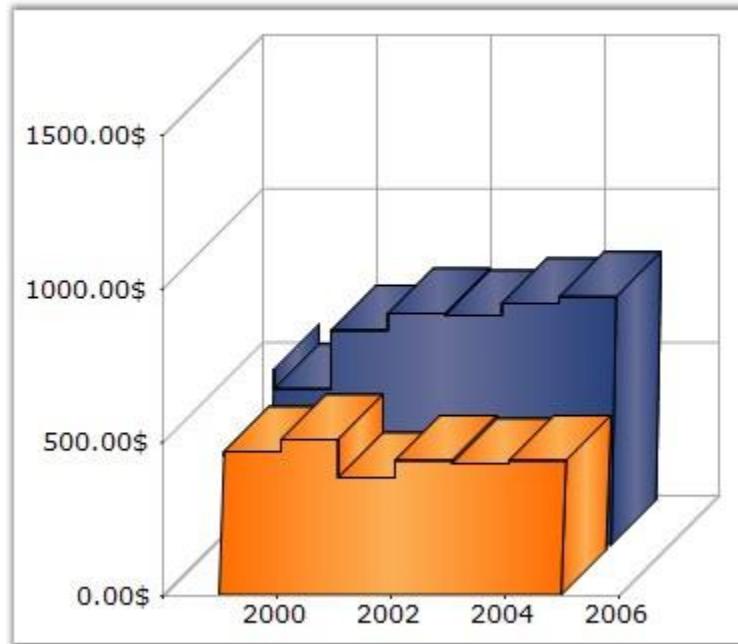
Here is sample code snippet using Inverted Step.

[C#]

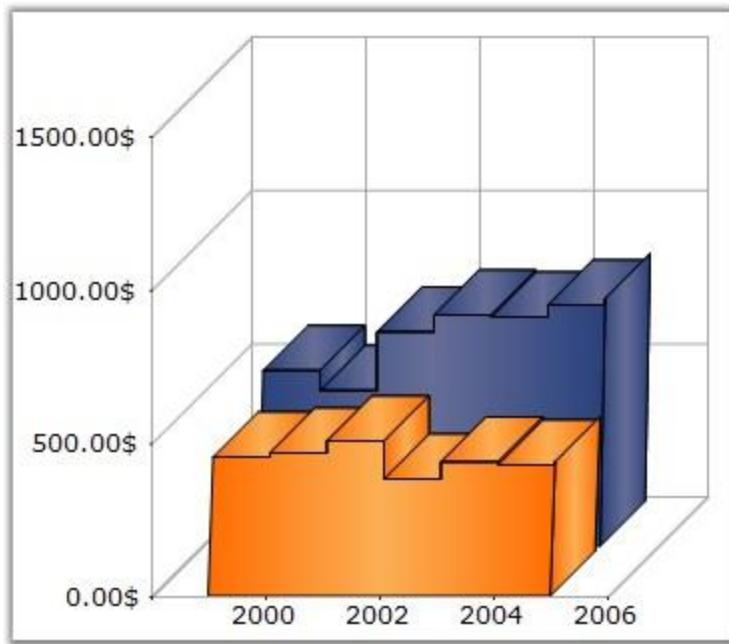
```
this.chartControl1.Series[0].ConfigItems.StepItem.Inverted=true;
```

**[VB .NET]**

```
Private Me.chartControl1.Series(0).ConfigItems.StepItem.Inverted=True
```



*Figure 211: Inverted = "False"*



*Figure 212: Inverted = "True"*

#### See Also

[StepAreaChart](#), [StepLine Chart](#)

#### 4.5.1.79 Summary

Provides access to summary information such as minimum/maximum values contained in this series at any given moment.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"><li>• MaxX - Returns the maximum X value.</li><li>• MaxY - Returns the minimum Y value.</li><li>• MinX - Returns the minimum X value.</li><li>• MinY - Returns the minimum Y value.</li><li>• ModelImpl - Returns the model implemented.</li><li>• GetYPercentage - This method returns percentage value of series point in a Pie</li></ul>

	chart.
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>MaxX</b> - 0</li> <li>• <b>MaxY</b> - 0</li> <li>• <b>MinY</b> - 0</li> <li>• <b>MinX</b> - 0</li> <li>• <b>ModelImpl</b> - Nil</li> </ul>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All chart types

Here is a sample code snippet using Radar chart.

**[C#]**

```
string str = this.chartControl1.Series[0].Summary.MaxY.ToString();
string str1 = this.chartControl1.Series[0].Summary.MinY.ToString();
label1.Text = "Summary" + "\n" + " MaxY Value : " + str + "\n" + "MinY
Value : " + str1;

//To get percentage value of series point in Pie chart
this.chartControl1.Series[0].Summary.GetYPercentage(1);
this.chartControl1.Series[0].Summary.GetYPercentage(1, 0);
```

**[VB.NET]**

```
Dim str As String = Me.chartControl1.Series(0).Summary.MaxY.ToString()
Dim str1 As String = Me.chartControl1.Series(0).Summary.MinY.ToString()
label1.Text = "Summary" + "" & Chr(10) & "" + " MaxY Value : " + str +
"" & Chr(10) & "" + "MinY Value : " + str1

'To get percentage value of series point in Pie chart
this.chartControl1.Series[0].Summary.GetYPercentage(1)
this.chartControl1.Series[0].Summary.GetYPercentage(1, 0)
```

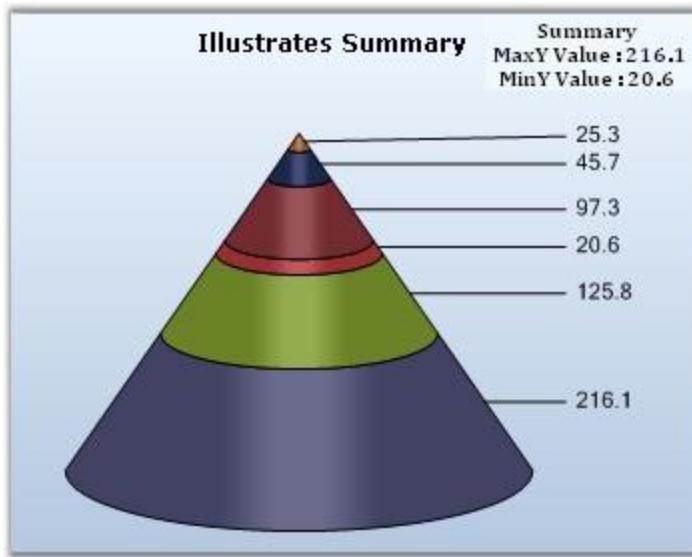


Figure 213: Summary values displayed in Labels

#### See Also

[Chart Types](#), [How to find value, maximum value and minimum value among the data points](#)

#### 4.5.1.80 Symbol

Sets the attributes of the symbol that is to be displayed at this point.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>Size</b> - Specifies the size of the symbol.</li> <li>• <b>Color</b> - Specifies the color of the symbol.</li> <li>• <b>Shape</b> - Specifies the shape of the symbol.</li> <li>• <b>Offset</b> - Specifies the offset of the symbol.</li> </ul>
<b>Default Value</b>	<ul style="list-style-type: none"> <li>• <b>Size</b> - 10,10</li> <li>• <b>Color</b> - HighLightTextColor</li> <li>• <b>Shape</b> - None</li> </ul>

	• <b>Offset - 0,0</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Column Chart, Bar Chart, Bubble Chart, Financial Chart, Line Chart, BoxandWhisker Chart, Gantt chart, Tornado chart, Radar Chart, Polar Chart, Area Charts, Scatter Chart

Here is sample code snippet using Symbol in Column Chart.

### Series Wide Setting

#### [C#]

```
this.chartControl1.Series[0].Style.Symbol.Shape =
ChartSymbolShape.Diamond;
this.chartControl1.Series[0].Style.Symbol.Color=Color.Green;
this.chartControl1.Series[0].Style.Symbol.Size = new Size(10, 10);
this.chartControl1.Series[0].Style.Symbol.Offset = new Size(1, 20);

//Sets the Color of the Symbol border.
this.chartControl1.Series[0].Style.Symbol.Border.Color = Color.Blue;
//Sets the Width of the Symbol border.
this.chartControl1.Series[0].Style.Symbol.Border.Width = 1;
//Sets the Alignment of the Symbol border.
this.chartControl1.Series[0].Style.Symbol.Border.Alignment =
PenAlignment.Outset;
//Sets the Dash style of the Symbol Border.
this.chartControl1.Series[0].Style.Symbol.Border.DashStyle =
DashStyle.Solid;
```

#### [VB .NET]

```
Private Me.chartControl1.Series(0).Style.Symbol.Shape =
ChartSymbolShape.Diamond
Private Me.chartControl1.Series(0).Style.Symbol.Color=Color.Green
Private Me.chartControl1.Series(0).Style.Symbol.Size = New Size(10, 10)
Private Me.chartControl1.Series(0).Style.Symbol.Offset = New Size(1,
20)

'Used to set the Color of the Symbol border.
```

```
Private Me.chartControl1.Series(0).Style.Symbol.Border.Color =  
Color.Blue  
'Used to set the Width of the Symbol border.  
Private Me.chartControl1.Series(0).Style.Symbol.Border.Width = 1  
'Used to set the Alignment of the Symbol border.  
Private Me.chartControl1.Series(0).Style.Symbol.Border.Alignment =  
PenAlignment.Outlet  
'Used to set the Dash style of the Symbol Border.  
Private Me.chartControl1.Series0).Style.Symbol.Border.DashStyle =  
DashStyle.Solid
```

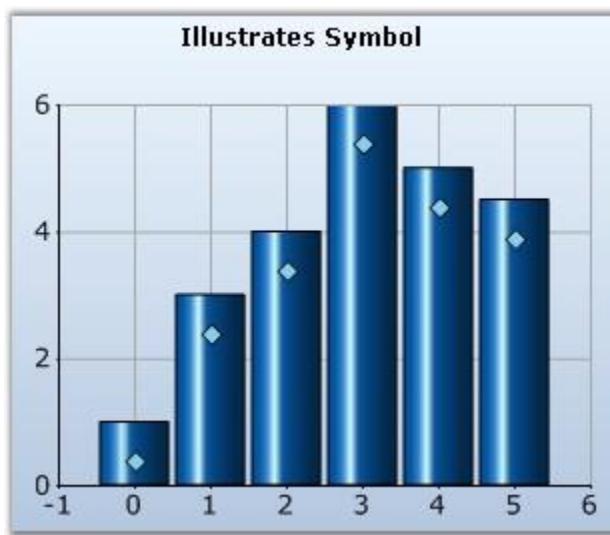


Figure 214: Blue Color Diamond symbol in a Column Chart with vertical offset of 20

### Specific Data Point Setting

To specify customized symbols for individual datapoints, use **Series.Styles[i].Symbol** property, where i ranges from 0 to n representing the data points.

```
[C#]  
  
this.chartControl1.Series[0].Styles[0].Symbol.Shape =  
ChartSymbolShape.Diamond;  
this.chartControl1.Series[0].Styles[0].Symbol.Color=Color.Green;  
this.chartControl1.Series[0].Styles[0].Symbol.Size = new Size(10, 10);  
this.chartControl1.Series[0].Styles[0].Symbol.Offset = new Size(1, 20);  
  
this.chartControl1.Series[0].Styles[1].Symbol.Shape =  
ChartSymbolShape.Hexagon;  
this.chartControl1.Series[0].Styles[1].Symbol.Color=Color.Yellow;
```

```
this.chartControl1.Series[0].Styles[1].Symbol.Size = new Size(9, 9);
this.chartControl1.Series[0].Styles[1].Symbol.Offset = new Size(1, 20);

//Used to set the Color of the Symbol border.
this.chartControl1.Series[0].Styles[0].Symbol.Border.Color =
Color.Blue;
//Used to set the Width of the Symbol border.
this.chartControl1.Series[0].Styles[0].Symbol.Border.Width = 3;
//Used to set the Alignment of the Symbol border.
this.chartControl1.Series[0].Styles[0].Symbol.Border.Alignment =
PenAlignment.Outset;
//Used to set the Dash style of the Symbol Border.
this.chartControl1.Series[0].Styles[0].Symbol.Border.DashStyle =
DashStyle.Solid;
```

**[VB.NET]**

```
Private Me.chartControl1.Series(0).Styles(0).Symbol.Shape =
ChartSymbolShape.Diamond
Private Me.chartControl1.Series(0).Styles(0).Symbol.Color=Color.Green
Private Me.chartControl1.Series(0).Styles(0).Symbol.Size = New Size(10,
10)
Private Me.chartControl1.Series(0).Styles(0).Symbol.Offset = New
Size(1, 20)

Private Me.chartControl1.Series(0).Styles(1).Symbol.Shape =
ChartSymbolShape.Hexagon
Private Me.chartControl1.Series(0).Styles(1).Symbol.Color=Color.Yellow
Private Me.chartControl1.Series(0).Styles(1).Symbol.Size = New Size(9,
9)
Private Me.chartControl1.Series(0).Styles(1).Symbol.Offset = New
Size(1, 20)

'Used to set the Color of the Symbol border.
Private Me.chartControl1.Series(0).Styles[0].Symbol.Border.Color =
Color.Blue
'Used to set the Width of the Symbol border.
Private Me.chartControl1.Series(0).Styles[0].Symbol.Border.Width = 3
'Used to set the Alignment of the Symbol border.
Private Me.chartControl1.Series(0).Styles[0].Symbol.Border.Alignment =
PenAlignment.Outset
'Used to set the Dash style of the Symbol Border.
Private Me.chartControl1.Series0).Styles[0].Symbol.Border.DashStyle =
DashStyle.Solid
```

**See Also**

[Column Charts](#), [Bar Charts](#), [Bubble Chart](#), [Financial Chart](#), [Line Charts](#), [BoxandWhisker Chart](#), [Gantt chart](#), [Tornado chart](#), [Radar Chart](#), [Polar Chart](#), [Area Charts](#), [Scatter Chart](#)

#### 4.5.1.81 Text (Series)

It specifies the chart series text. This will be displayed at run-time in the legend.

Details	
<b>Possible Values</b>	Any string value
<b>Default Value</b>	<b>Name of the series</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All Chart Types

Text can be set directly by using Series object.

Here is sample code snippet.

[C#]

```
// Here the series text will be taken from series name
ChartSeries series1 =
this.chartControl1.Model.NewSeries("August", ChartSeriesType.Column);

// Add points to series1.
//....
//....

// Here, the text is given explicitly.

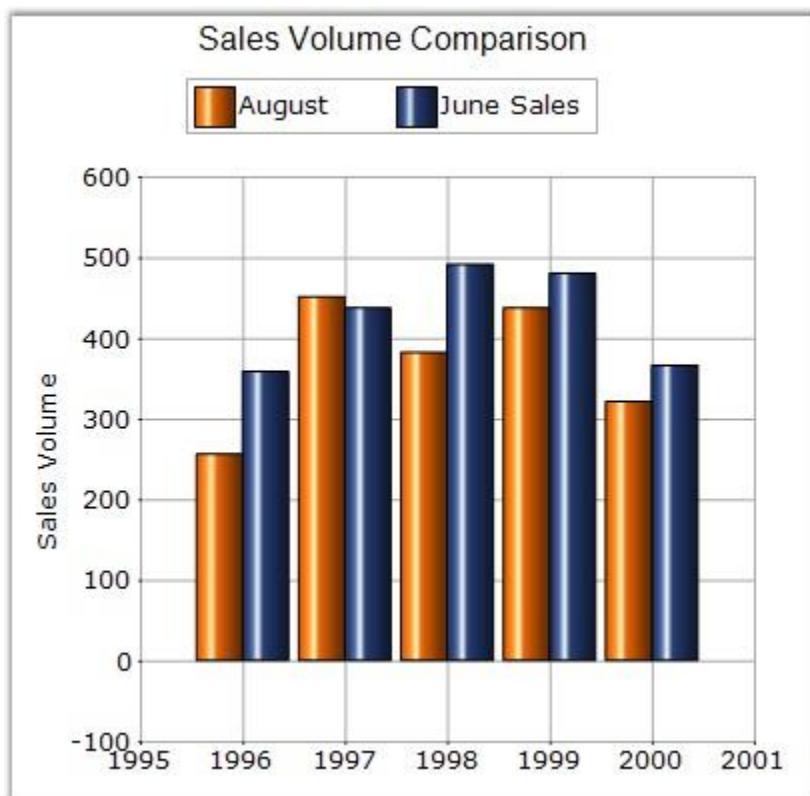
ChartSeries series2 =
this.chartControl1.Model.NewSeries("June", ChartSeriesType.Column);
series2.Text = "JuneSales";
```

[VB.NET]

```
' Here the series text will be taken from series name
Dim series1 As ChartSeries =
Me.chartControll.Model.NewSeries("August",ChartSeriesType.Column)

' Add points to series1.
'....
' Here, the text is given explicitly.

Dim series2 As ChartSeries =
Me.chartControll.Model.NewSeries("June",ChartSeriesType.Column)
series2.Text = "JuneSales"
```



*Figure 215: Text set for the Series*

**See Also**

[Chart Types](#)

#### 4.5.1.82 Text (Style)

##### Series Wide Setting

Datapoint labels for a series can be specified using **Series.Style.Text** property.

[C#]

```
//labels for the series  
chartControl1.Series[0].Style.DisplayText = true;  
chartControl1.Series[0].Style.Text = "Series1 Point";
```

[VB .NET]

```
'labels for the series  
chartControl1.Series[0].Style.DisplayText = True  
chartControl1.Series(0).Style.Text = "Series1 Point"
```

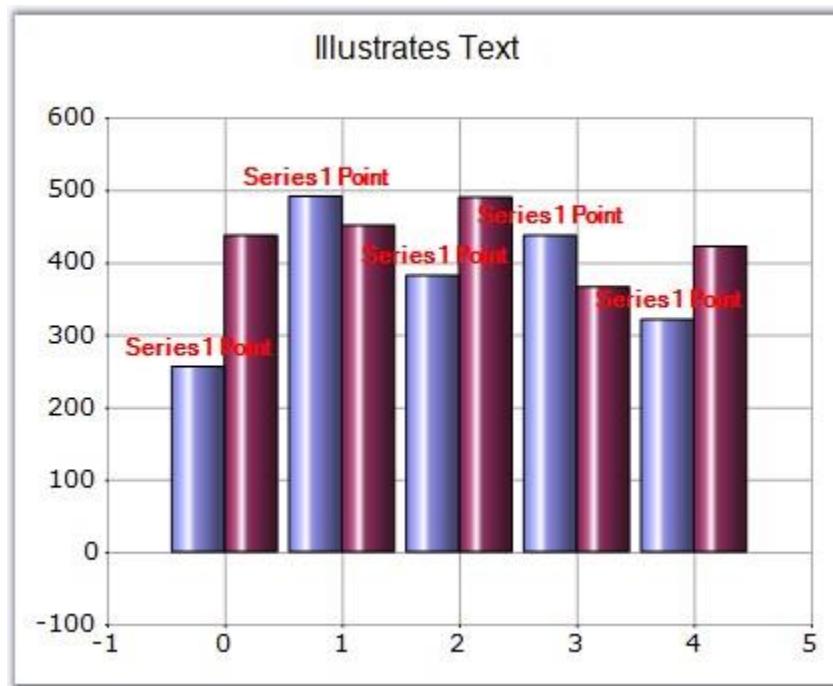


Figure 216: DataPoint Labels displayed for the Series

##### Specific Data Point Setting

Labels for specific data points can be specified through **Series.Styles[0].Text** property.

**[C#]**

```
//labels for the individual datapoints in the series  
chartControl1.Series[0].Style.DisplayText = true;  
chartControl1.Series[0].Styles[0].Text = "First Point";  
chartControl1.Series[0].Styles[1].Text = "Second Point";  
chartControl1.Series[0].Styles[2].Text = "Third Point";
```

**[VB .NET]**

```
'labels for the individual datapoints in the series  
chartControl1.Series[0].Style.DisplayText = True  
chartControl1.Series(0).Styles(0).Text = "First Point"  
chartControl1.Series(0).Styles(1).Text = "Second Point"  
chartControl1.Series(0).Styles(2).Text = "Third Point"
```

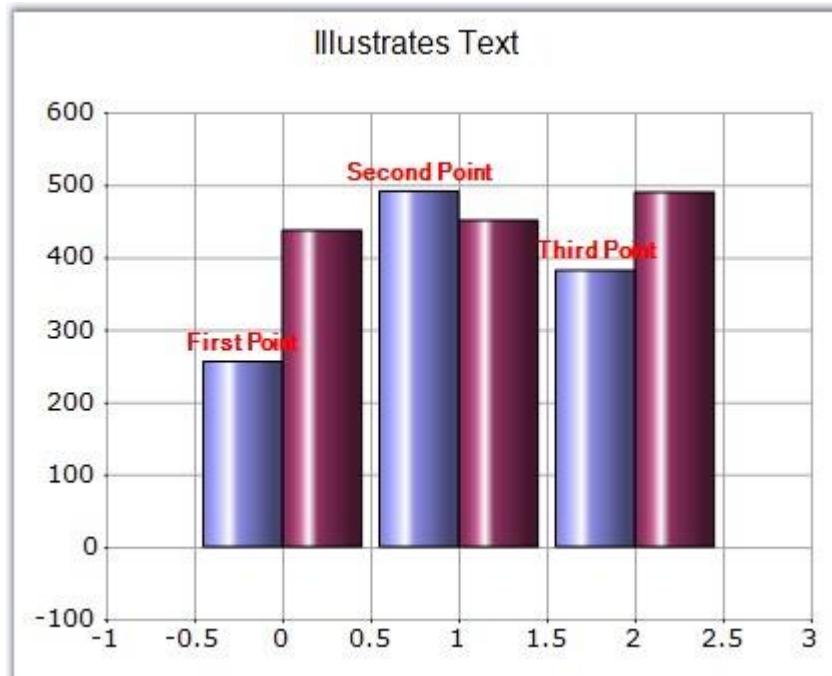
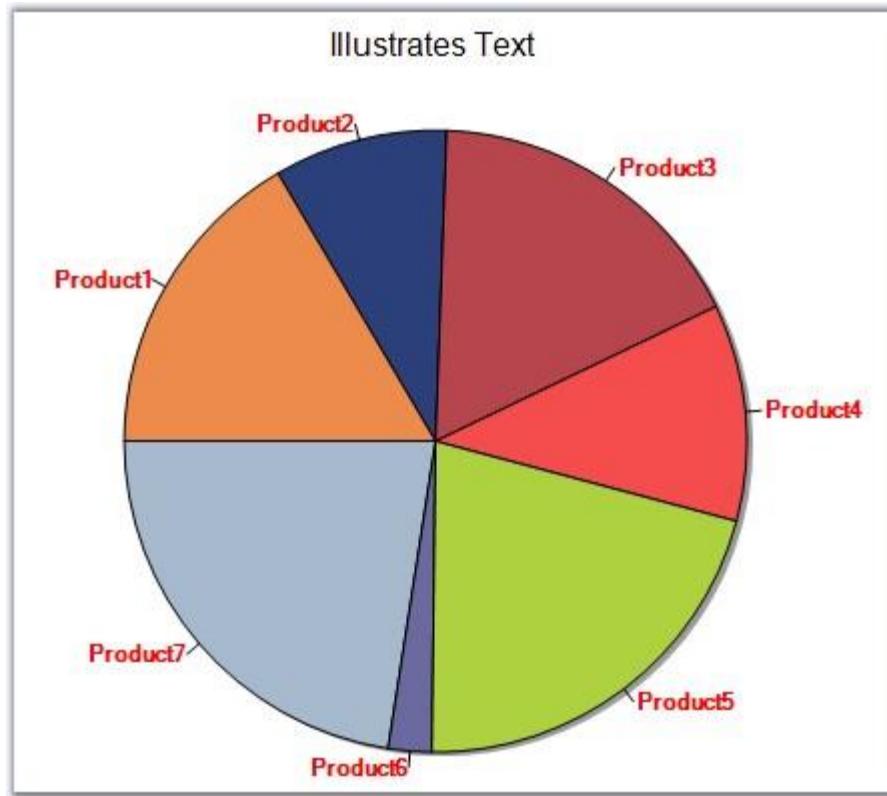


Figure 217: Using Series.Styles[0].Text in Column Chart



*Figure 218: Using Series.Styles[0].Text in Pie Chart*

#### **See Also**

[Chart Types](#)

#### **4.5.1.83 TextColor**

It is used to set the color of the display text of the series.

Details	
Possible Values	All predefined colors
Default Value	Black
2D / 3D Limitations	No
Applies to Chart Element	Series and Points

**Applies to Chart Types**

All Chart Types

Here is sample code snippet using `TextColor` in Column Chart.

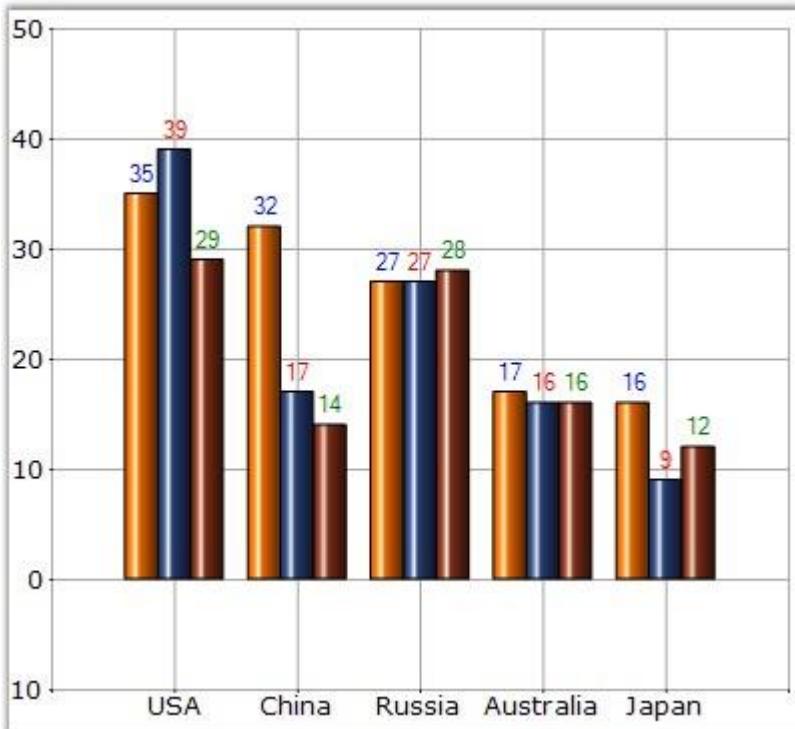
### **Series Wide Setting**

**[C#]**

```
// Set the color of the text in the Series  
this.chartControl1.Series[0].Style.TextColor = Color.Blue;  
this.chartControl1.Series[1].Style.TextColor = Color.Red;  
this.chartControl1.Series[2].Style.TextColor = Color.Green;
```

**[VB .NET]**

```
' Set the color of the text in the Series  
Private Me.chartControl1.Series(0).Style.TextColor = Color.Blue  
Private Me.chartControl1.Series(1).Style.TextColor = Color.Red  
Private Me.chartControl1.Series(2).Style.TextColor = Color.Green
```



*Figure 219: Customized Series Text Color*

## Specific Data Point Setting

We can set TextColor for specific data points in a series by using **Series.Styles[0].TextColor** property as follows.

### [C#]

```
// Set the text color for the three data points in the Series  
this.chartControl1.Series[0].Styles[0].TextColor = Color.Blue;  
this.chartControl1.Series[0].Styles[1].TextColor = Color.SteelBlue;  
this.chartControl1.Series[0].Styles[2].TextColor = Color.LightBlue;
```

### [VB .NET]

```
' Set the text color for the three data points in the Series  
Private Me.chartControl1.Series(0).Styles(0).TextColor = Color.Blue  
Private Me.chartControl1.Series(0).Styles(1).TextColor =  
Color.SteelBlue  
Private Me.chartControl1.Series(0).Styles(2).TextColor =  
Color.LightBlue
```

## See Also

[Chart Types](#)

### 4.5.1.84 TextFormat

Sets the format that is to be applied to values that are displayed as text

Details	
<b>Possible Values</b>	Any string value with '{0}' as place holder for the Y value.
<b>Default Value</b>	<b>Nil</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series and Points
<b>Applies to Chart Types</b>	All Chart Types

Here is sample code snippet using TextFormat in Column Chart.

### **Series wide setting**

[C#]

```
this.chartControl1.Series[0].Style.TextFormat = "T = {0}";
```

[VB .NET]

```
Me.chartControl1.Series(0).Style.TextFormat = "T = {0}"
```

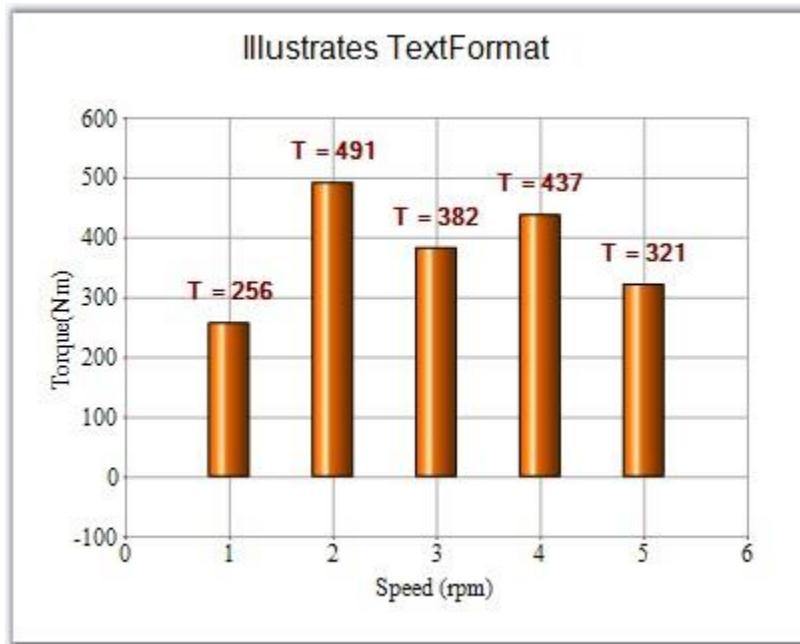


Figure 220: *TextFormat = "T = {0}"*

### **Specific Data Point Setting**

TextFormats for individual data points are specified using below code.

[C#]

```
chartControl1.Series[0].Styles[0].TextFormat = "YValue : {0}";  
chartControl1.Series[0].Styles[1].TextFormat = "Dollars : {0:C}";
```

**[VB .NET]**

```
chartControl1.Series(0).Styles(0).TextFormat = "YValue : {0}"
chartControl1.Series(0).Styles(1).TextFormat = "Dollars : {0:C}"
```

**See Also**

[Chart Types](#)

#### 4.5.1.85 TextOffset

Sets the Offset of the text from the position of the chart point.

Details	
<b>Possible Values</b>	Any float value
<b>Default Value</b>	2.5
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All Chart Types

Here is a sample code snippet using TextOffset in Column Chart.

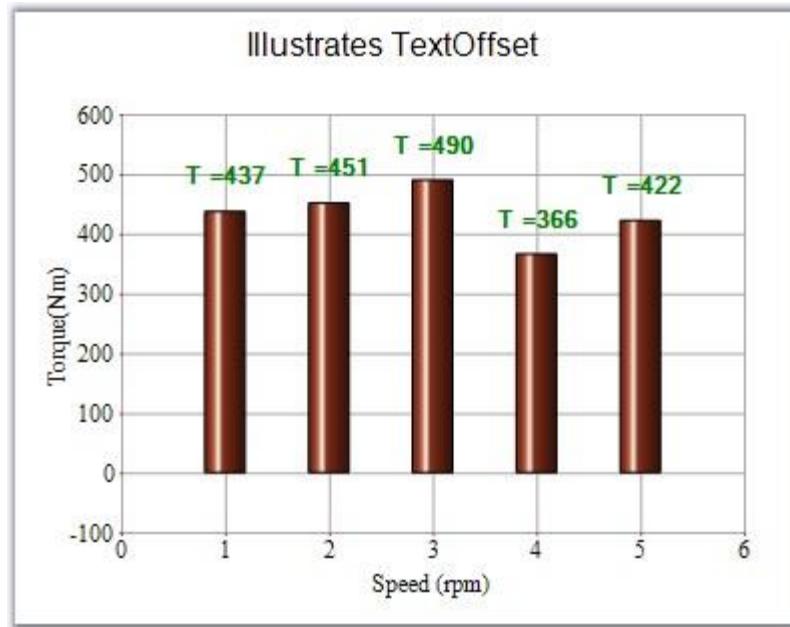
##### Series Wide Setting

**[C#]**

```
this.chartControl1.Series[0].Style.TextOffset = 10.0F;
```

**[VB .NET]**

```
Me.chartControl1.Series(0).Style.TextOffset = 10.0F
```



*Figure 221: Text Offset set to 10.0F*

### Specific Data Point Setting

TextOffset for data points can be specified using **Series.Styles[0].TextOffset** property.

#### [C#]

```
this.chartControl1.Series[0].Styles[0].TextOffset = 10.0F;  
this.chartControl1.Series[0].Styles[1].TextOffset = 15.0F;
```

#### [VB .NET]

```
Me.chartControl1.Series(0).Styles(0).TextOffset = 10.0F  
Me.chartControl1.Series(0).Styles(1).TextOffset = 15.0F
```

### See Also

[Chart Types](#)

### 4.5.1.86 TextOrientation

It is used to align the text of the series within the data point region.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>Center</b> - Aligns to the center of the point.</li> <li>• <b>Down</b> - Aligns below the point.</li> <li>• <b>Left</b> - Aligns to the left position to the point.</li> <li>• <b>RegionCenter</b> - Aligns below the region that represents the points.</li> <li>• <b>RegionDown</b> - Aligns below the region that represents the points.</li> <li>• <b>RegionUp</b> - Aligns to the top of the region that represents the points.</li> <li>• <b>Right</b> - Aligns to the right of the point.</li> <li>• <b>Smart</b> - Aligns in a manner that is appropriate to the situation.</li> <li>• <b>SymbolCenter</b> - Text is centered to the symbol that is associated to the point.</li> <li>• <b>Up</b> - Aligns to the top of the point.</li> <li>• <b>UpLeft</b> - Aligns to the top left corner of the point.</li> <li>• <b>UpRight</b> - Aligns to the top right corner of the point.</li> </ul>
<b>Default Value</b>	<b>Up</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All chart types

Here is some sample code.

#### Series Wide Setting

[C#]

```
// Text Orientation of chart series  
this.chartControl1.Series[1].Style.TextOrientation =  
ChartTextOrientation.RegionDown;  
this.chartControl1.Series[0].Style.TextOrientation =  
ChartTextOrientation.Up;  
this.chartControl1.Series[0].Style.TextColor=Color.Blue;  
this.chartControl1.Series[1].Style.TextColor=Color.Red;
```

[VB.NET]

```
' Text Orientation of chart series  
Private Me.chartControl1.Series(1).Style.TextOrientation =  
ChartTextOrientation.RegionDown  
Private Me.chartControl1.Series(0).Style.TextOrientation =  
ChartTextOrientation.Up  
Private Me.chartControl1.Series(0).Style.TextColor=Color.Blue  
Private Me.chartControl1.Series(1).Style.TextColor=Color.Red
```

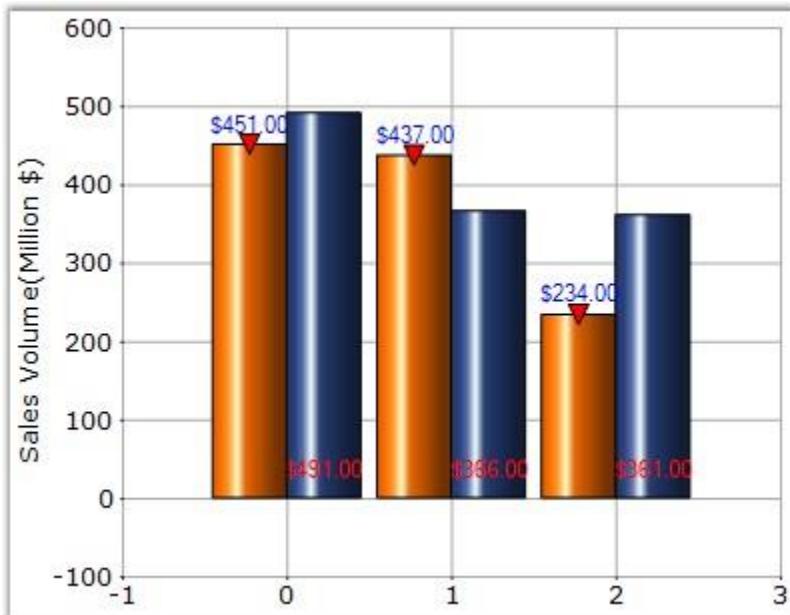


Figure 222: Text Orientation set for the Chart Series

### Specific Data Point Setting

Text orientation for specific data points can be set using **Series.Style[i].TextOrientation** property, where "i" represents the index of data points ranging from 0 to n.

**[C#]**

```
this.chartControl1.Series[1].Styles[0].TextOrientation =
ChartTextOrientation.RegionDown;
this.chartControl1.Series[0].Styles[0].TextOrientation =
ChartTextOrientation.Up;
this.chartControl1.Series[0].Styles[0].TextColor=Color.Blue;
this.chartControl1.Series[1].Styles[0].TextColor=Color.Red;

this.chartControl1.Series[1].Styles[1].TextOrientation =
ChartTextOrientation.Smart;
this.chartControl1.Series[0].Styles[1].TextOrientation =
ChartTextOrientation.UpRight;
this.chartControl1.Series[0].Styles[1].TextColor=Color.Green;
this.chartControl1.Series[1].Styles[1].TextColor=Color.Yellow;
```

**[VB.NET]**

```
Private Me.chartControl1.Series(1).Styles(0).TextOrientation =
ChartTextOrientation.RegionDown
Private Me.chartControl1.Series(0).Styles(0).TextOrientation =
ChartTextOrientation.Up
Private Me.chartControl1.Series(0).Styles(0).TextColor=Color.Blue
Private Me.chartControl1.Series(1).Styles(0).TextColor=Color.Red

Private Me.chartControl1.Series(1).Styles(1).TextOrientation =
ChartTextOrientation.Smart
Private Me.chartControl1.Series(0).Styles(1).TextOrientation =
ChartTextOrientation.UpRight
Private Me.chartControl1.Series(0).Styles(1).TextColor=Color.Green
Private Me.chartControl1.Series(1).Styles(1).TextColor=Color.Yellow
```

**See Also**

[Chart Types](#)

#### 4.5.1.87 ToolTip

Sets the tooltip of the style object associated with the series.

**Details**

<b>Possible Values</b>	Any string value
<b>Default Value</b>	<b>Nil</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Scatter Chart, Hilo Open Close Chart(3D), Column Charts, BarCharts, Bubble Chart, Line Charts, BoxandWhisker Chart, Tornado Chart, Combination Chart, Gantt Chart, Candle Chart, HiLo Chart(3D), PolarAndRadar, PieChart, Accumulation Charts, Area Charts

## Property

Table 1: Property Table

Property	Description	Type	Data Type	Reference links
ShowToolTips	Specifies whether tooltip has to be displayed.	Server side	Boolean	NA

Here is sample code snippet using ToolTip in the Column Chart.

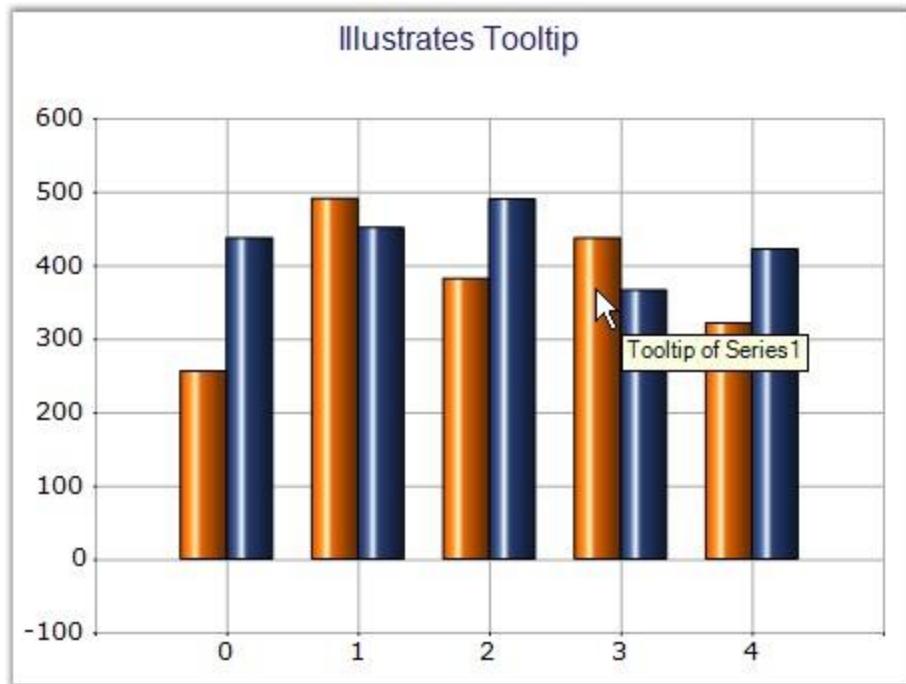
### Series Wide Setting

[C#]

```
this.chartControl1.ShowToolTips = true;
series1.PointsToolTipFormat = "{1}";
series1.Style.ToolTip = "Tooltip of Series1";
```

[VB .NET]

```
Me.chartControl1.ShowToolTips = True
series1.PointsToolTipFormat = "{1}"
series1.Style.ToolTip = "Tooltip of Series1"
```



*Figure 223: ToolTip set for Chart Series*

### **Specific Data Point Setting**

ToolTip can be applied to individual points of a Series.

[C#]

```
for (int i = 0; i < series1.Points.Count; i++)
{
    series1Styles[i].ToolTip = string.Format("X = {0}, Y = {1}",
series1.Points[0].X.ToString(),
series1.Points[i].YValues[0]);
}
```

[VB]

```
Dim i As Integer = 0
Do While i < series1.Points.Count
    series1Styles(i).ToolTip = String.Format("X = {0}, Y = {1}",
series1.Points(0).X.ToString(), series1.Points(i).YValues(0))
    i += 1
Loop
```

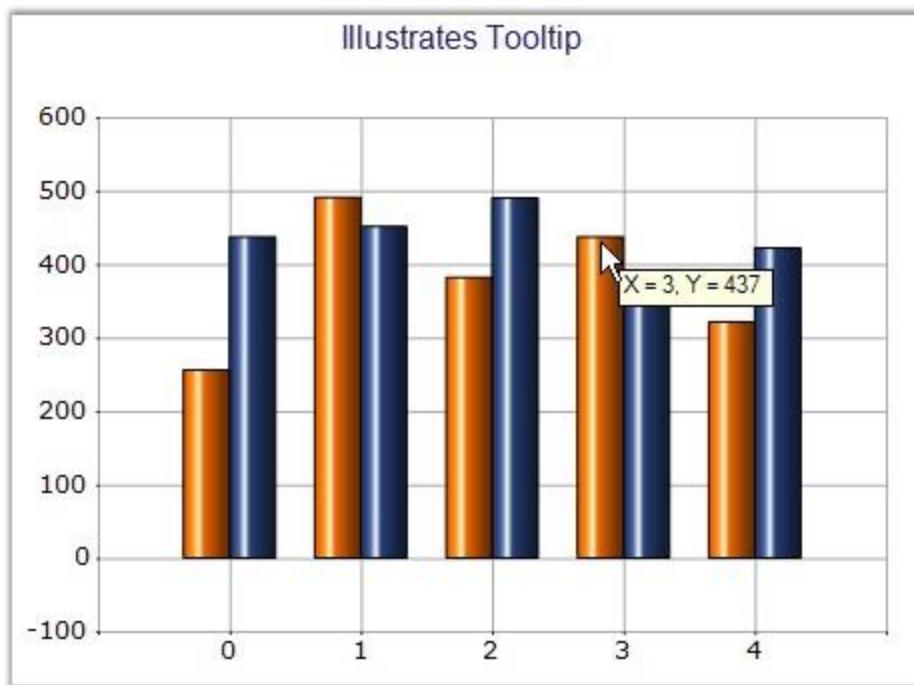


Figure 224: ToolTip set for Data Point in the Series

**See Also**

[Scatter Chart](#), [Hilo Open Close Chart\(3D\)](#), [Column Charts](#), [BarCharts](#), [Bubble Chart](#), [Line Charts](#), [BoxandWhisker Chart](#), [Tornado Chart](#), [Combination Chart](#), [Gantt Chart](#), [Candle Chart](#), [HiLo Chart\(3D\)](#), [PolarAndRadar chart](#), [PieChart](#), [Accumulation Charts](#), [Area Charts](#)

#### 4.5.1.88 ToolTipFormat

Sets the tooltip format of the style object associated with the series.

Details	
<b>Possible Values</b>	Any string with '{0}' as the place holder for Y value.
<b>Default Value</b>	Nil
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series and Points

<b>Applies to Chart Types</b>	Scatter Chart, Hilo Open Close Chart(3D), Column Charts, BarCharts, Bubble Chart, Line Charts, BoxandWhisker Chart, Tornado Chart, Combination Chart, Gantt Chart, Candle Chart, HiLo Chart(3D), PolarAndRadar, PieChart, Accumulation Charts, Area Charts
-------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Here is sample code snippet using ToolTipFormat in the Column chart.

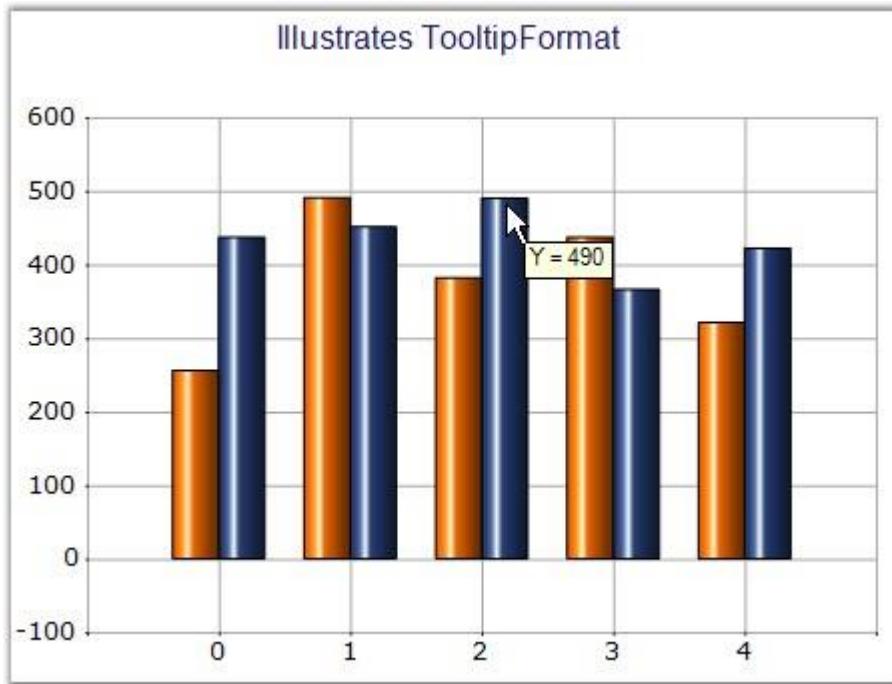
#### **Series Wide Setting**

**[C#]**

```
this.chartControl1.ShowToolTips = true;  
this.chartControl1.Series[1].Style.ToolTipFormat = "Y = {0}";
```

**[VB .NET]**

```
Me.chartControl1.ShowToolTips = True  
Me.chartControl1.Series[1].Style.ToolTipFormat = "Y = {0}"
```



*Figure 225: ToolTipFormat set for Chart Series*

## Specific Data Point Setting

ToolTipformat can be applied for individual points by using **Series.Styles[0].ToolTipFormat** property settings.

[C#]

```
for (int i = 0; i < series1.Points.Count; i++)
{
    series1.Styles[i].ToolTipFormat = "{0}";
}
```

[VB]

```
Dim i As Integer = 0
Do While i < series1.Points.Count
    series1.Styles(i).ToolTipFormat = "{0}";
    i += 1
Loop
```

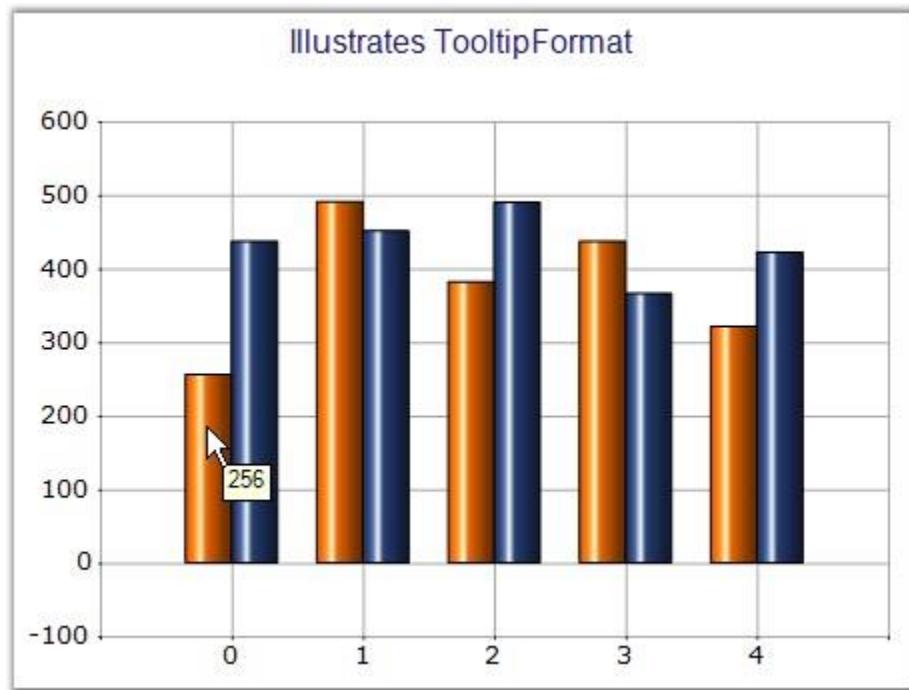


Figure 226: ToolTip Format set for Data Point in a Series

## See Also

[Scatter Chart](#), [Hilo Open Close Chart\(3D\)](#)

#### 4.5.1.89 Visible

It turns on / off the visibility of the series.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"><li>• <b>True</b> - Unhides the associated series.</li><li>• <b>False</b> - Hides the associated series.</li></ul>
<b>Default Value</b>	<b>True</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	All series
<b>Applies to Chart Types</b>	All chart types

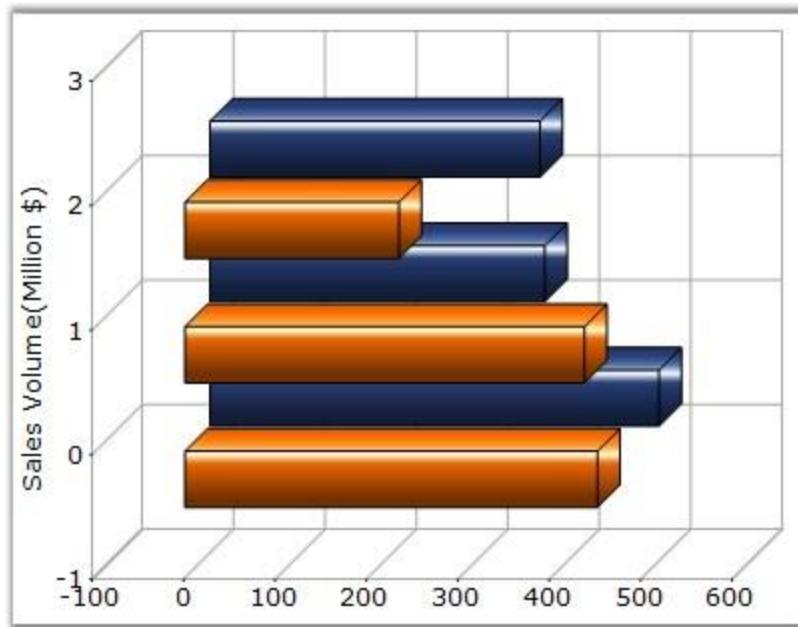
Here is sample code snippet using **Visible** property in Bar Chart.

[C#]

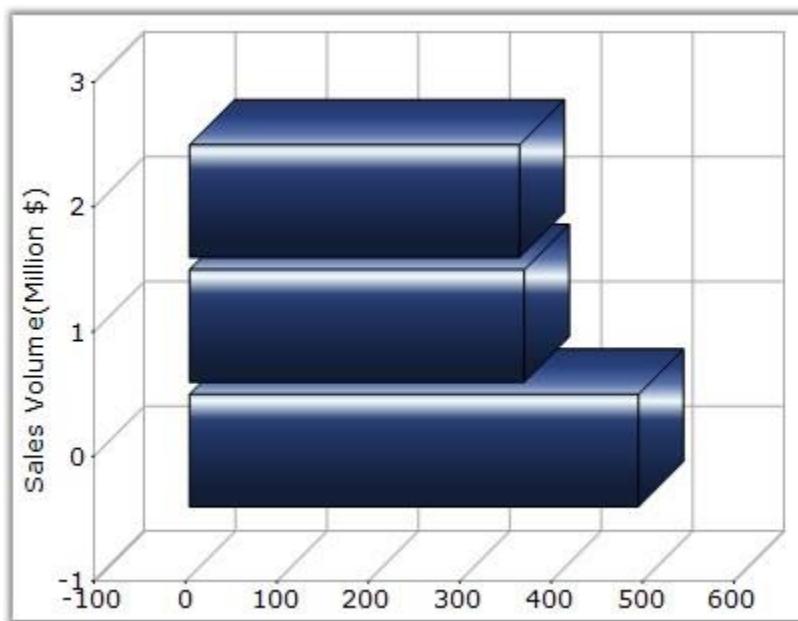
```
// Hides Series[0] and shows Series[1]
this.chartControl1.Series[0].Visible = false;
this.chartControl1.Series[1].Visible = true;
```

[VB.NET]

```
' Hides Series[0] and shows Series[1]
Private Me.chartControl1.Series(0).Visible = False
Private Me.chartControl1.Series(1).Visible = True
```



*Figure 227: Second Series Visible*



*Figure 228: Second Series Hidden*

## **See Also**

[Chart Types](#)

#### 4.5.1.90 VisibleAllPies

Specifies whether the legend is to be displayed with one legend item for each slice in the pie.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>True</b> - Indicates only one legend item for all slices of pie.</li> <li>• <b>False</b> - Indicates one legend item for each slice of pie.</li> </ul>
<b>Default Value</b>	<b>False</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	Pie Chart

Here is the sample code snippet using VisibleAllPies in PieChart.

[C#]

```
this.chartControl1.ChartArea.VisibleAllPies = false;
chartControl1.Legend.RowsCount = 3;
```

[VB .NET]

```
Me.chartControl1.ChartArea.VisibleAllPies = False
chartControl1.Legend.RowsCount = 3
```

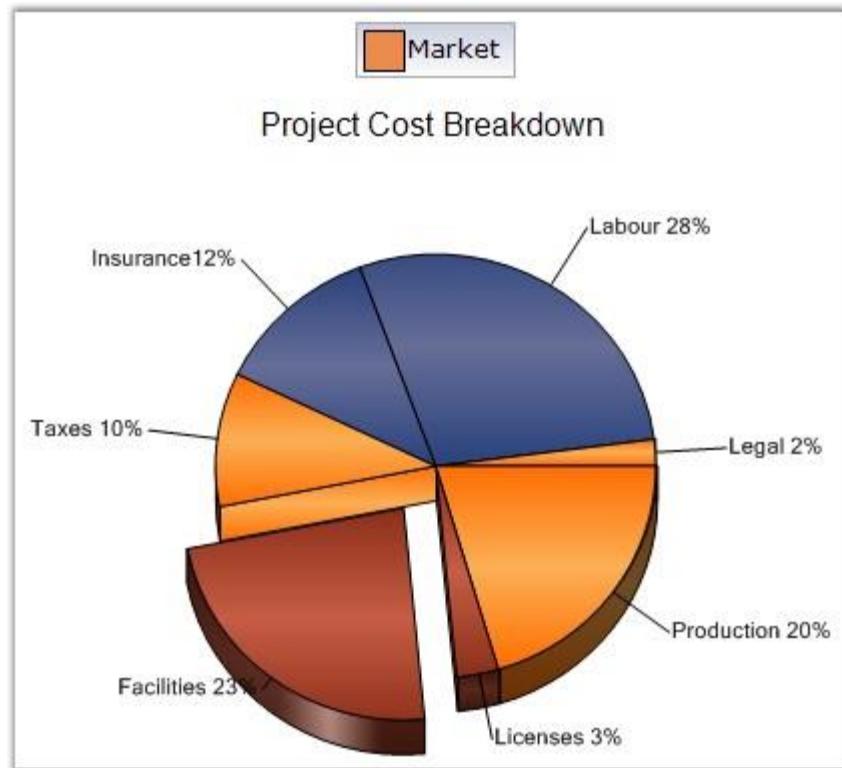


Figure 229: *VisibleAllPies* set to *True*

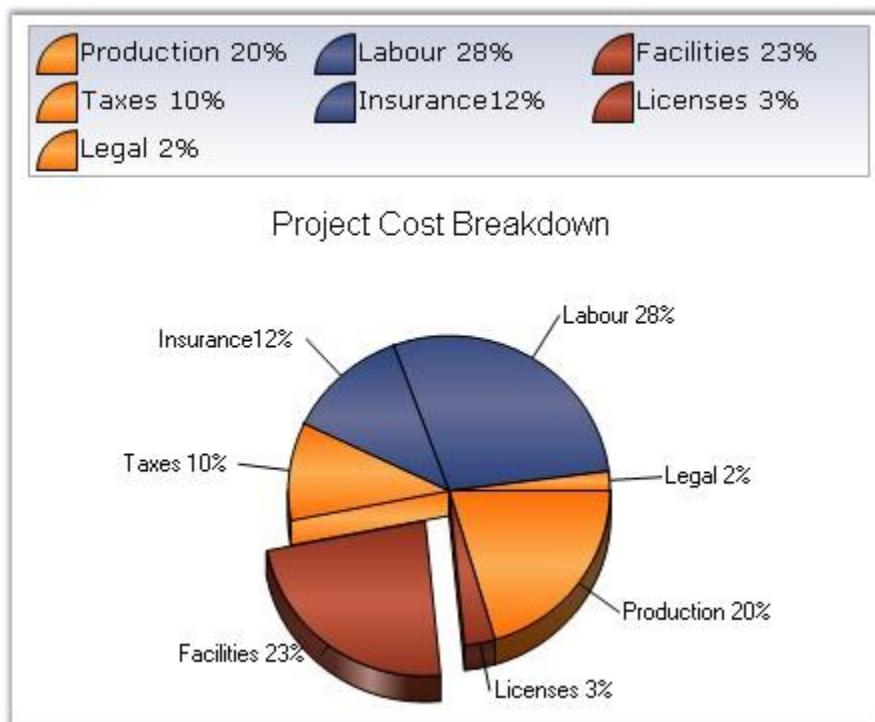


Figure 230: VisibleAllPies set to False

**See Also**

[Pie Chart](#)

#### 4.5.1.91 XType

Returns the x value type that is being rendered. It is a read-only property.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li>• <b>Double</b> - Specifies Double values</li> <li>• <b>DateTime</b> - Specifies Date and Time values</li> <li>• <b>Logarithmic</b> - Specifies Logarithmic values</li> <li>• <b>Custom</b> - Specifies Custom values</li> </ul>
<b>Default Value</b>	<b>Double</b>
<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All Chart Types

Here is sample code snippet using XType.

**[C#]**

```
autoLabel1.Text = this.chartControl1.Series[0].XType.ToString();
```

**[VB .NET]**

```
Private autoLabel1.Text = Me.chartControl1.Series(0).XType.ToString()
```



Figure 231: Get the X Value Type that is being Rendered

## See Also

[Chart Types](#)

### 4.5.1.92 YType

Returns the y value type that is being rendered. It is a read-only property.

Details	
<b>Possible Values</b>	<ul style="list-style-type: none"> <li><b>Double</b> - Specifies Double values</li> <li><b>DateTime</b> - Specifies Date and Time values</li> <li><b>Logarithmic</b> - Specifies Logarithmic values</li> <li><b>Custom</b> - Specifies Custom values</li> </ul>
<b>Default Value</b>	<b>Double</b>

<b>2D / 3D Limitations</b>	No
<b>Applies to Chart Element</b>	Any Series
<b>Applies to Chart Types</b>	All Chart Types

Here is sample code snippet using YType.

[C#]

```
autoLabel1.Text = this.chartControl1.Series[0].YType.ToString();
```

[VB .NET]

```
Private autoLabel1.Text = Me.chartControl1.Series(0).YType.ToString()
```



*Figure 232: Get the Y Value Type that is being Rendered*

## See Also

[Chart Types](#)

#### 4.5.1.93 ZOrder

Specifies the order in which the objects are arranged and controls the visibility when one is placed over the other.

By default, the ZOrder for series are assigned based on the order in which they are added to the Series collection.

Details	
Possible Values	Any integer value
Default Value	The order that we add the series in the chart control.
2D / 3D Limitations	No
Applies to Chart Element	Any Series
Applies to Chart Types	Gantt Chart, Histogram chart, Tornado Chart, Combination Chart, Box and Whisker Chart, Area Charts, Polar And Radar Chart, BarCharts, Column Charts, Bubble Charts, Candle Charts, Hilo Charts, Hilo Open Close Chart

Here is sample code snippet using ZOrder.

**[C#]**

```
this.chartControl1.Series[0].ZOrder = 0;
this.chartControl1.Series[1].ZOrder = 1;
```

**[VB .NET]**

```
Private Me.chartControl1.Series(0).ZOrder = 0
Private Me.chartControl1.Series(1).ZOrder = 1
```

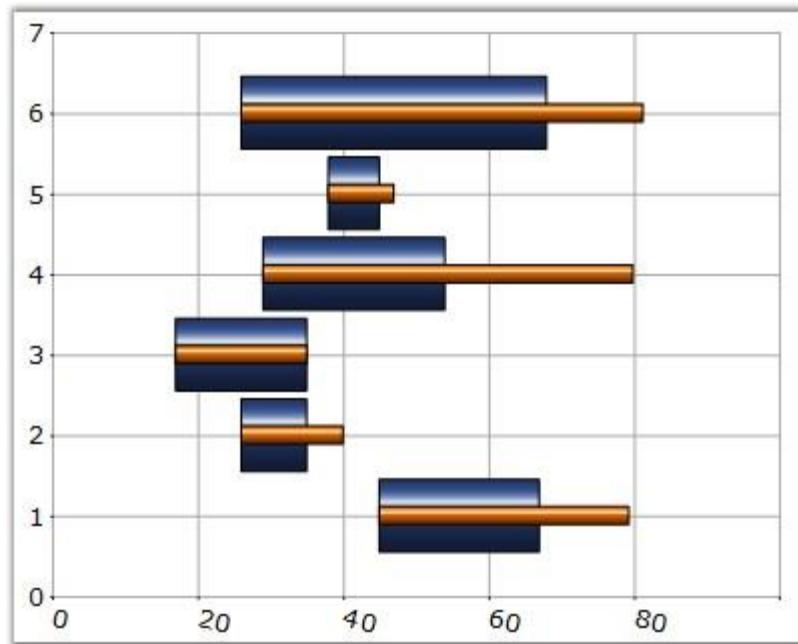


Figure 233: Series 1 ZOrder = 0, Series 2 ZOrder = 1

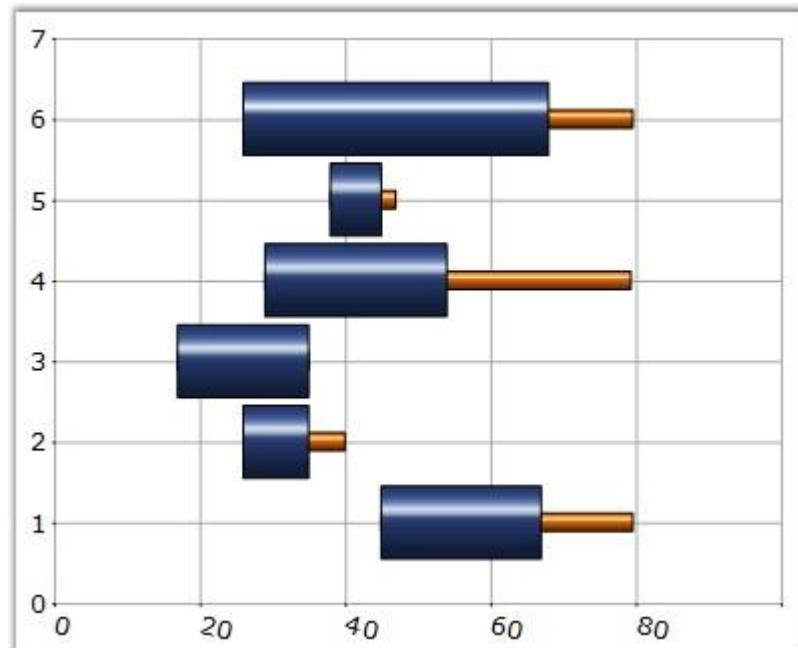


Figure 234: Series 1 ZOrder = 1, Series 2 ZOrder = 0

### Rearranging the Series using ZOrder property

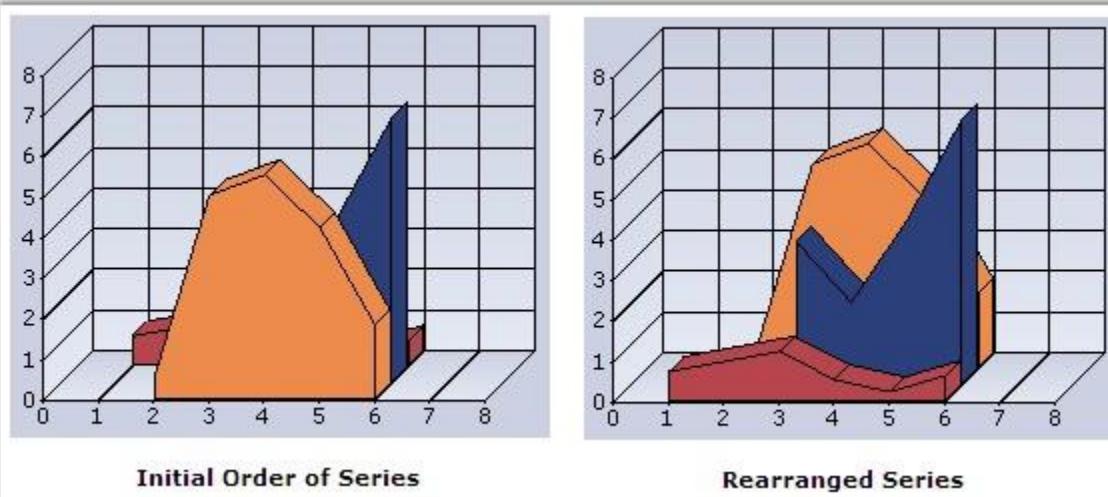
The chart series can be rearranged at run-time using ZOrder property as follows. The chart needs to be redrawn in order to reflect ZOrder property changes. we cannot call redrawing for every series ZOrder changes. In order to overcome this, we should change the order of the series in between the begin update and end update statements as follows.

**[C#]**

```
this.chartControl1.BeginUpdate();
this.chartControl1.Model.Series[0].ZOrder = 2;
this.chartControl1.Model.Series[1].ZOrder = 1;
this.chartControl1.Model.Series[2].ZOrder = 0;
this.chartControl1.EndUpdate();
```

**[VB .NET]**

```
Me.chartControl1.BeginUpdate()
Me.chartControl1.Model.Series[0].ZOrder = 2
Me.chartControl1.Model.Series[1].ZOrder = 1
Me.chartControl1.Model.Series[2].ZOrder = 0
Me.chartControl1.EndUpdate()
```



*Figure 235: Chart with Rearranged Series*

**See Also**

[Gantt Chart](#), [Histogram chart](#), [Tornado Chart](#), [Combination Chart](#), [Box and Whisker Chart](#), [Area Charts](#), [Polar And Radar Chart](#), [BarCharts](#), [Column Charts](#), [Bubble Charts](#), [Candle Charts](#), [Hilo Charts](#), [Hilo Open Close Chart](#)

## 4.5.2 Data Point Labels, Tooltips and Symbols

### Labels

Data Points in a series can be adorned with text labels as well as custom symbols to provide additional information regarding the specific data points.

Text labels can be rendered at the data points using the [DisplayText](#), [Text](#) and [TextFormat](#) settings. They can further be customized using the [TextColor](#), [TextOffset](#) and [TextOrientation](#) settings.

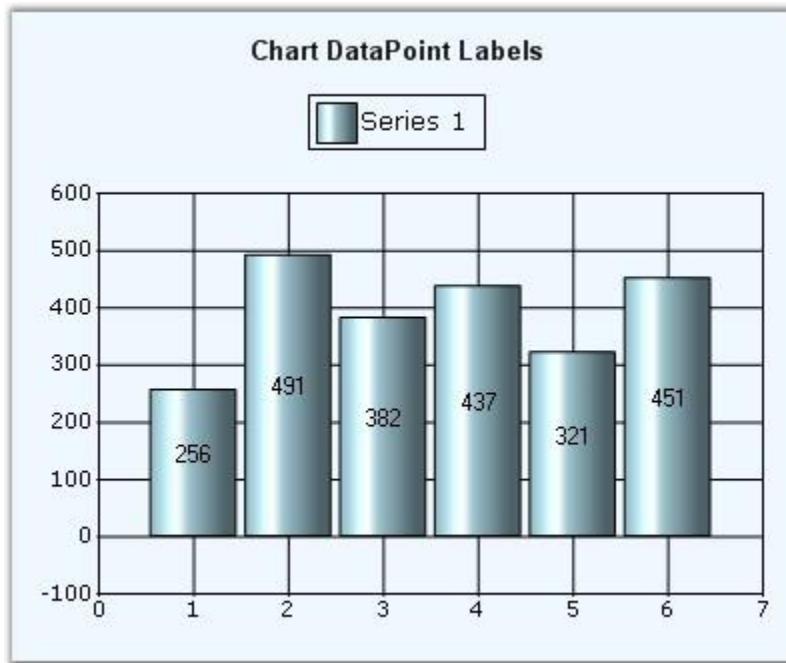


Figure 236: Black Color DataPoint Labels with TextOrientation= "RegionCenter", TextFormat="{0}"

### Tooltips

Refer the [Tooltips](#) topic for more information on this.

### Symbols

Built-in or custom symbols can be rendered at the data points to emphasize importance of certain data points. See [Symbol](#) setting for more information.

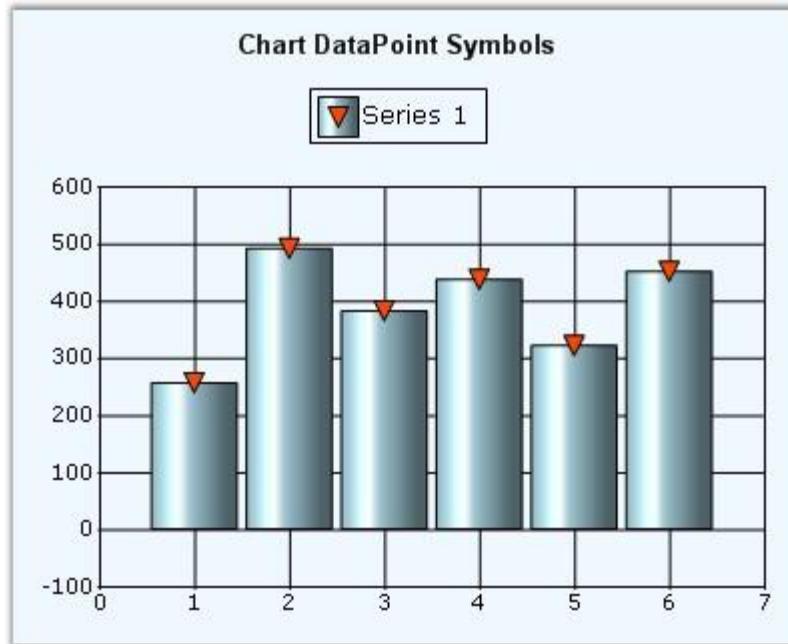


Figure 237: "OrangeRed InvertedTriangle" Symbols in the Series Points

### 4.5.3 Custom Points

Essential Chart supports plotting of points on the Chart Area even if they don't belong to a series. These are stored in the **ChartControl.CustomPoints** collection. They can be set at custom coordinates of the Chart Area or be made to follow a certain point or percentage coordinates. A custom point displays a text, background, border, symbol and marker, which is a line that connects the CustomPoint with the point on the chart area when it is offset from it.

Through Designer the Custom Points can be set using the **CustomPoints** property. Clicking this property will popup ChartCustomPoint Collection Editor window where you can add your custom points.

You can set the co-ordinates (**XValue** and the **YValue** property), symbols and their customization, using the **Symbols** property, text, using the **Text** property, alignment of the text, using the **Alignment** property and so on.

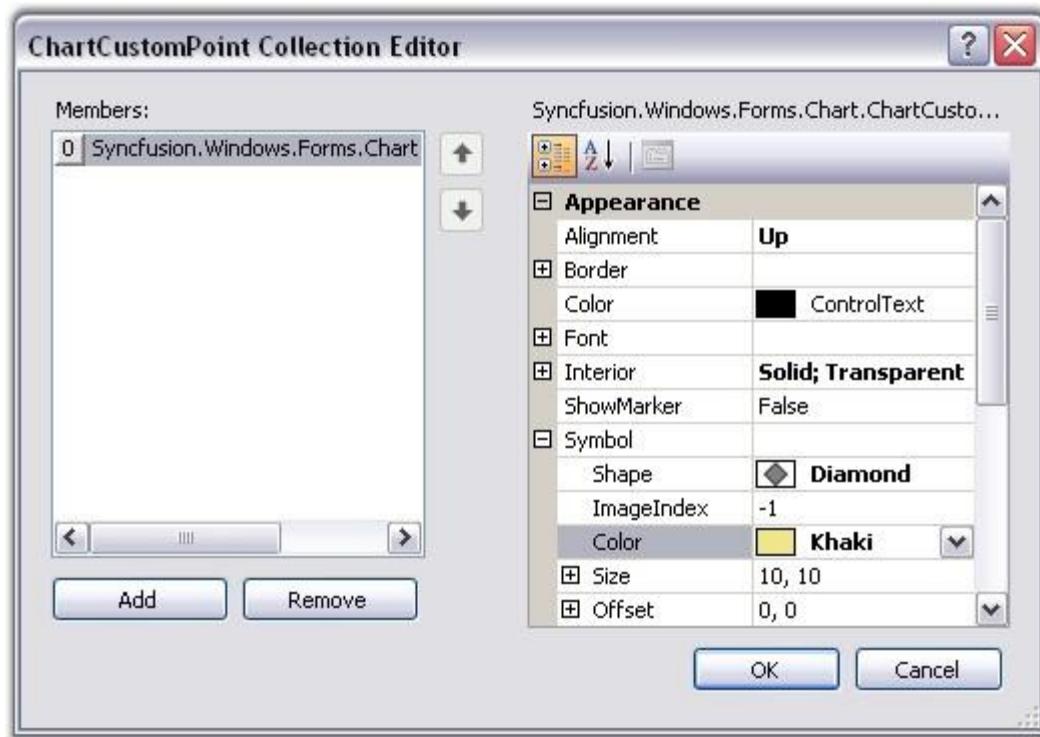


Figure 238: CustomPoints Collection Editor during Design Time

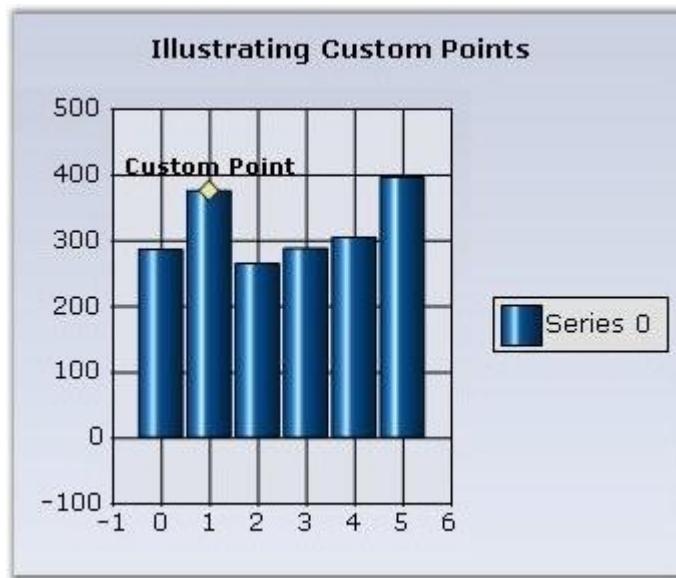


Figure 239: ChartControl with a Custom Point

### Programmatically

- Creating and Customizing the Custom Point.

**[C#]**

```
// Point that follows a series point:  
ChartCustomPoint cp = new ChartCustomPoint();  
  
// Gets the series index and point index if the Customtype is  
// Pointfollow.  
cp.PointIndex = 1;  
cp.SeriesIndex = 0;  
  
// Specifies the text of the custom point.  
cp.Text = "Custom Point";  
  
// Specifies the custom type.  
chartCustomPoint1.CustomType =  
Syncfusion.Windows.Forms.Chart.ChartCustomPointType.PointFollow;  
  
// Specifies the shape of the custom point symbol.  
cp.Symbol.Shape = ChartSymbolShape.Diamond;  
  
// Specifies the color of the custom point.  
chartCustomPoint1.Symbol.Color = Color.Khaki;  
  
//Setting X-value and Y- value  
chartCustomPoint1.XValue = 1;  
chartCustomPoint1.YValue = 370;  
  
//Setting the font properties  
chartCustomPoint1.Color = System.Drawing.SystemColors.ButtonHighlight;  
chartCustomPoint1.Font.Bold = true;  
chartCustomPoint1.Font.Facename = "Verdana";  
chartCustomPoint1.Font.Size = 10F;
```

**[VB.NET]**

```
'Point that follows a series point:  
cp As ChartCustomPoint = New ChartCustomPoint()  
  
'Gets the series index and point index if the Customtype is  
'Pointfollow.  
cp.PointIndex = 1  
cp.SeriesIndex = 0  
  
'Specifies the text of the custom point.
```

```
cp.Text = "Custom Point"

'Specifies the custom type.
chartCustomPoint1.CustomType =
Syncfusion.Windows.Forms.Chart.ChartCustomPointType.PointFollow

'Specifies the shape of the custom point symbol.
cp.Symbol.Shape = ChartSymbolShape.Diamond

'Specifies the color of the custom point.
cp.Symbol.Color = Color.Khaki

//Setting X-value and Y- value
chartCustomPoint1.XValue = 1
chartCustomPoint1.YValue = 370

'Setting the font properties
chartCustomPoint1.Color = System.Drawing.SystemColors.ButtonHighlight
chartCustomPoint1.Font.Bold = true
chartCustomPoint1.Font.Facename = "Verdana"
chartCustomPoint1.Font.Size = 10F
```



**Note:** You can also customize a custom point symbol using **Symbol** property.

- Adding Custom Point to the Chart.

**[C#]**

```
// Adds the custom point to the collection.
this.chartControl1.CustomPoints.Add(cp);
```

**[VB .NET]**

```
'Adds the custom point to the collection.
Me.chartControl1.CustomPoints.Add(cp)
```

### Custom point types

CustomPoint types	Description
PointFollow	This custom point will follow the regular points of any series, to which it is assigned.

ChartCoordinates	This lets you render a point type at any location in the chart.
Percent	The coordinates are specified as the percentage of the chart area.
Pixel	The coordinates are specified to be in pixels of the chart area.

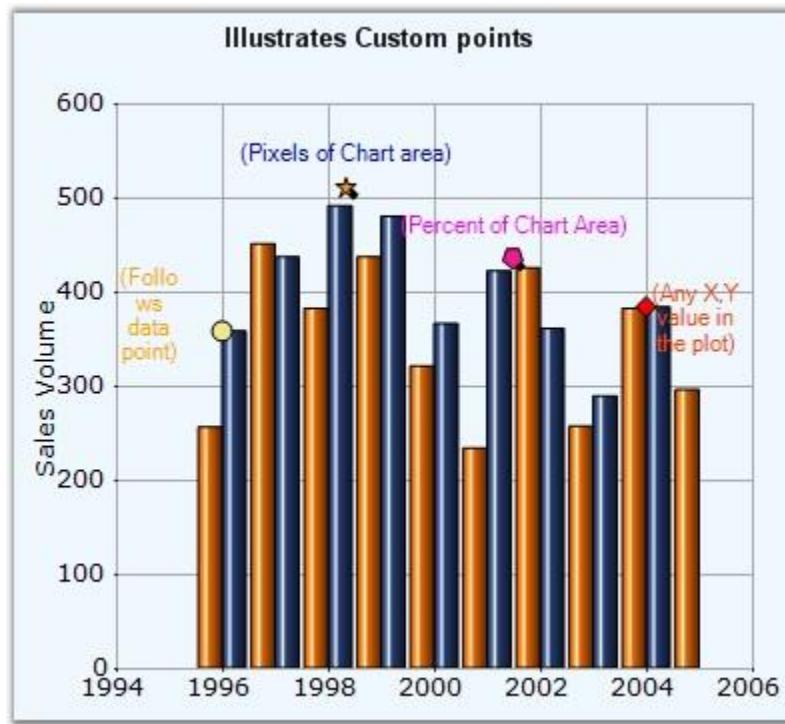


Figure 240: Custom Point Types Illustrated

The custom point symbols in the above image represents following Custom Types respectively.

1. Yellow “Circle” – PointFollow
2. Orange “Star” - Pixel
3. Pink “Pentagon” - Percent
4. OrangeRed “Diamond” - ChartCoordinates

A sample demonstrating all the custom point types is available in our installation at the following location:

<..\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Chart Series\Chart Custom Points>

#### 4.5.3.1 Custom Point in Multiple Axes

The custom points for the Secondary axis can be achieved by assigning the Series Index for the ChartCoordinates type.

The following code snippet illustrates this:

```
[C#]  
ChartCustomPoint cp = new ChartCustomPoint();  
cp.CustomType = ChartCustomPointType.ChartCoordinates;  
//Set the series index if the Customtype is ChartCoordinates in  
multiple axis  
cp.SeriesIndex = 0;  
cp.XValue = 10;  
cp.YValue = 60;  
cp.Symbol.Shape = ChartSymbolShape.Circle;  
cp.Alignment = ChartTextOrientation.Left;  
cp.Color = Color.Black;  
cp.Font.Facename = "Verdana";  
cp.Font.Size = 8.0F;  
cp.Symbol.Color = Color.Yellow;  
cp.Text = cp.XValue + "," + cp.YValue;  
this.ChartWebControl1.CustomPoints.Add(cp);
```

```
[VB]  
Dim cp As ChartCustomPoint = New ChartCustomPoint()  
cp.CustomType = ChartCustomPointType.ChartCoordinates  
'Set the series index if the Customtype is ChartCoordinates in multiple  
axis  
cp.SeriesIndex = 0  
cp.XValue = 10  
cp.YValue = 60  
cp.Symbol.Shape = ChartSymbolShape.Circle  
cp.Alignment = ChartTextOrientation.Left  
cp.Color = Color.Black  
cp.Font.Facename = "Verdana"  
cp.Font.Size = 8.0F  
cp.Symbol.Color = Color.Yellow  
cp.Text = cp.XValue & "," & cp.YValue  
Me.ChartWebControl1.CustomPoints.Add(cp)
```

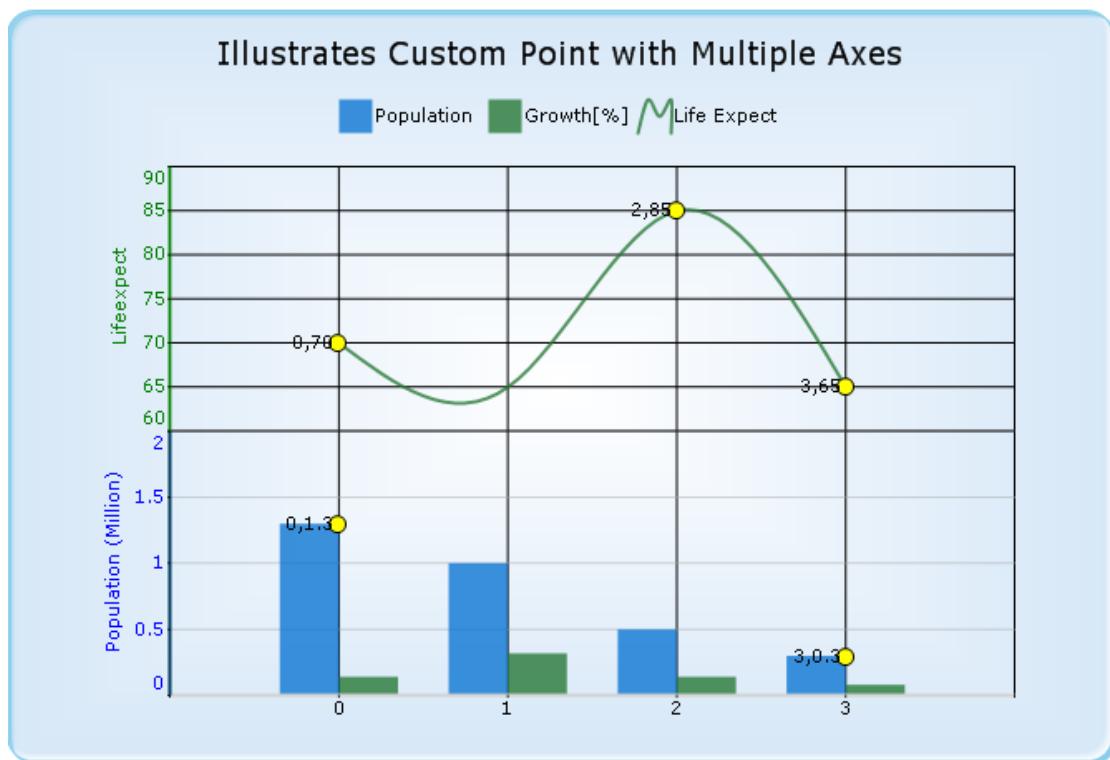


Figure 241: Custom Points in Multiple Axes

#### 4.5.4 Empty Points

Essential Chart lets you prevent certain points from getting plotted in the resultant chart. Such points are termed **Empty Points**.

Empty Points can be implemented by setting the **IsEmpty** property of the **ChartPoint** class to **true**.

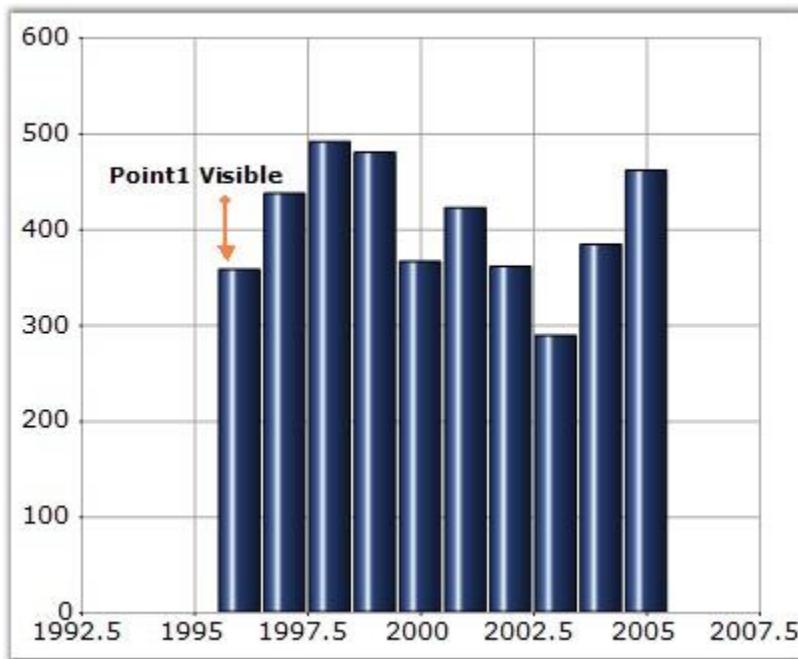
[C#]

```
// This sets the specified point as empty point.
this.chartControl1.Series[1].Points[0].IsEmpty = true;
```

**[VB.NET]**

```
' This sets the specified point as empty point.  
Me.chartControl1.Series[1].Points[0].IsEmpty = True
```

The following images illustrates the same. The second image displays after setting Point1 as an empty point.



*Figure 242: Chart without Empty Point*

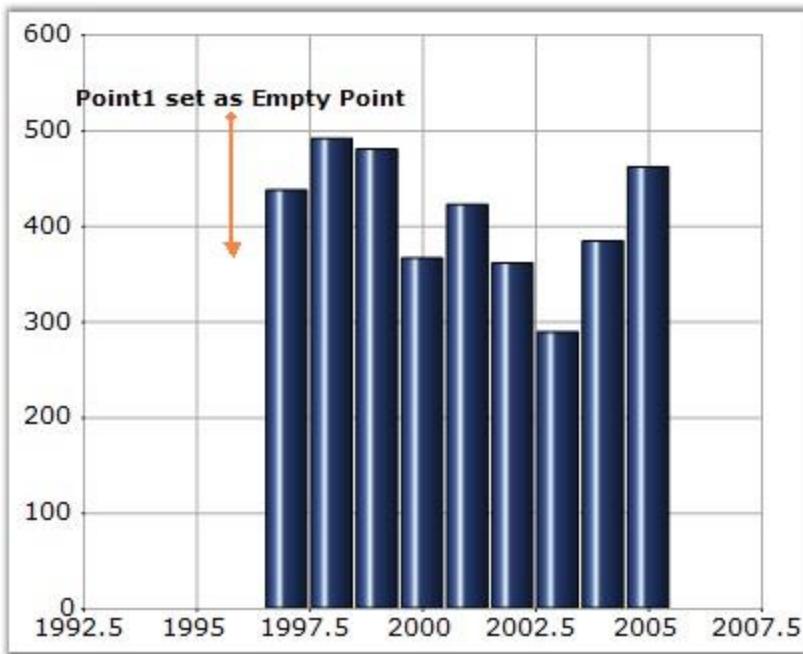


Figure 243: Chart with Point1 as Empty Point

### Showing Empty Point without any gap between Data Points

It is possible to set some data point as empty point and still show the chart without any gap between the points. You need to set **AllowGapForEmptyPoints** property to **false** to enable this feature. By default it is set to **true**.



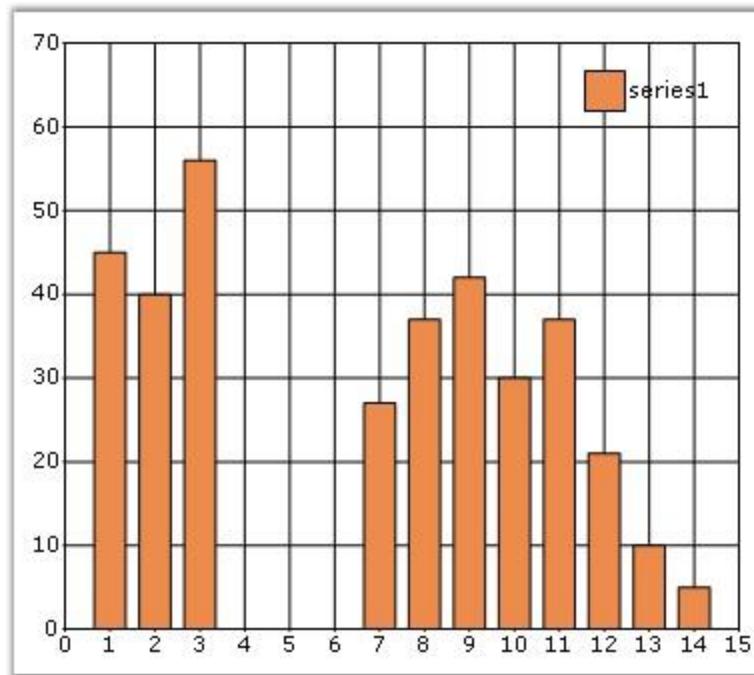
**Note:** You need to set **ChartControl.Indexed** property to **true** for the above setting to be effective.

#### [C#]

```
this.chartControl1.Series[0].Points[3].IsEmpty = true;
this.chartControl1.Series[0].Points[4].IsEmpty = true;
this.chartControl1.Series[0].Points[5].IsEmpty = true;
```

#### [VB .NET]

```
Me.chartControl1.Series(0).Points(3).IsEmpty = True
Me.chartControl1.Series(0).Points(4).IsEmpty = True
Me.chartControl1.Series(0).Points(5).IsEmpty = True
```



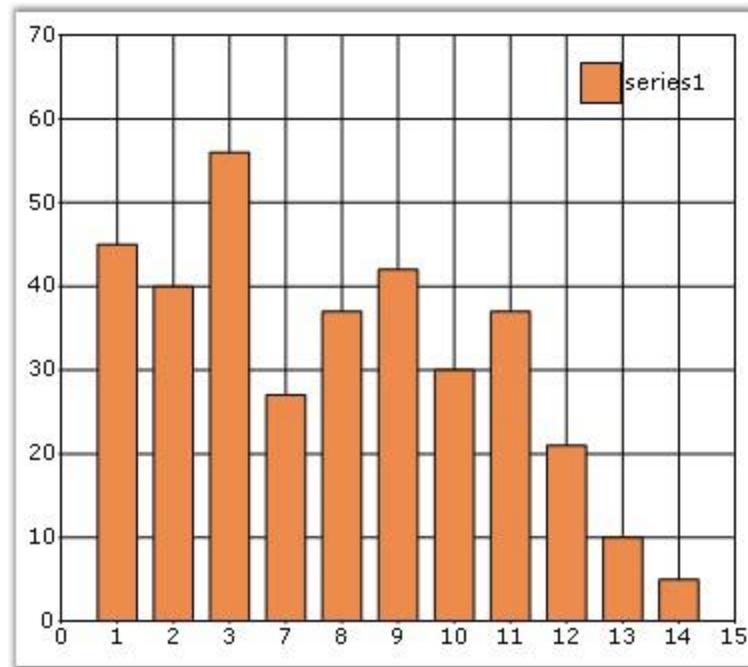
*Figure 244: 4th, 5th and 6th Data Points set as Empty Points*

**[C#]**

```
this.chartControl1.Indexed = true;  
this.chartControl1.AllowGapForEmptyPoints = true;
```

**[VB.NET]**

```
Me.chartControl1.Indexed = True  
Me.chartControl1.AllowGapForEmptyPoints = True
```



*Figure 245: 4th, 5th and 6th Data Points set as Empty Points; AllowGapForEmptyPoints = "True"*

## 4.6 Chart Axes

An axis in Essential Chart is represented by the **ChartAxis** class. The axes are stored in the Chart control's **Axes** collection.

By default, this collection contains two primary axes. These two axes can also be accessed through the **PrimaryXAxis** and **PrimaryYAxis**.

The PrimaryXAxis is usually rendered at the bottom, horizontally, while the PrimaryYAxis is usually rendered at the left, vertically. There might be exceptions to this rule as in the case of a Bar chart where the above x and y axis positions are reversed.

The title for an axis is set through the **Title** property, and the **TitleAlignment** property lets you align the same.

Axes features are discussed under the following topics:

#### 4.6.1 Indexed X Values

By default points in a series are plotted against their x and y values. However in some cases the x values are meaningless, they simply represent categories, and you do not want to plot the points against such x values. Such an x-axis that ignores the x-values and simply uses the positional value of a point in a series is said to be Indexed.

In the figure below, the first chart shows a line chart that is not-indexed while the second chart shows a line chart whose x-axis is indexed.

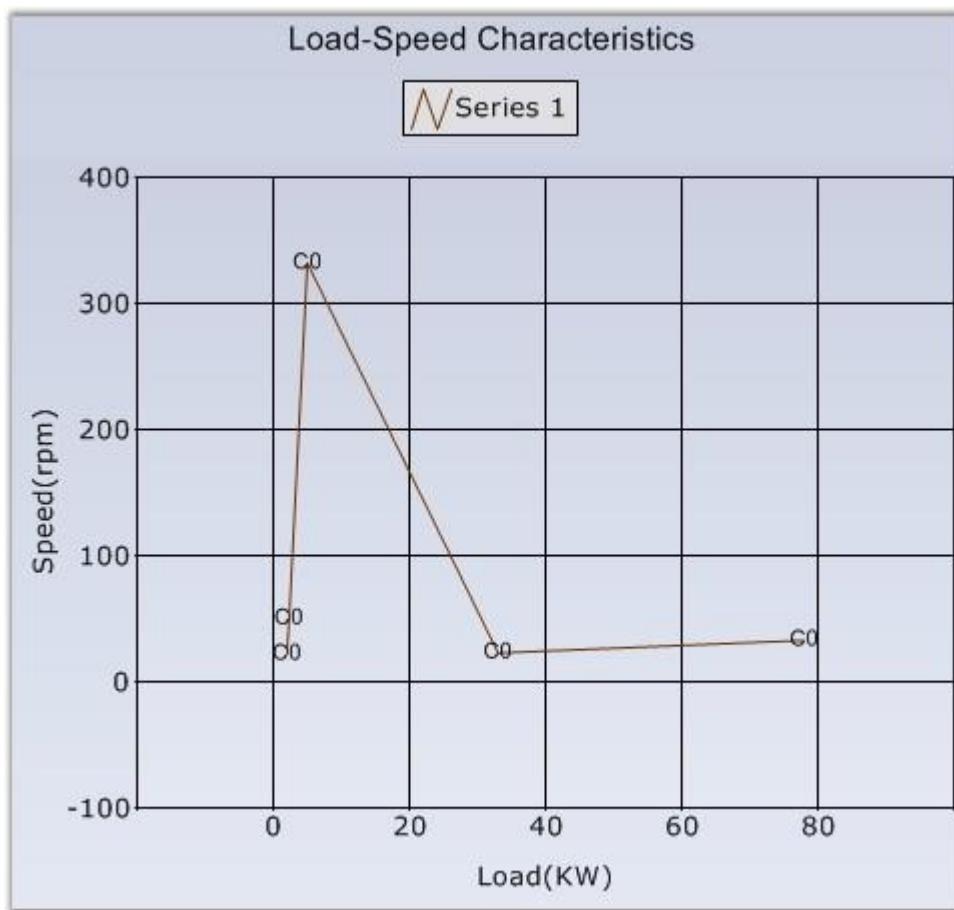
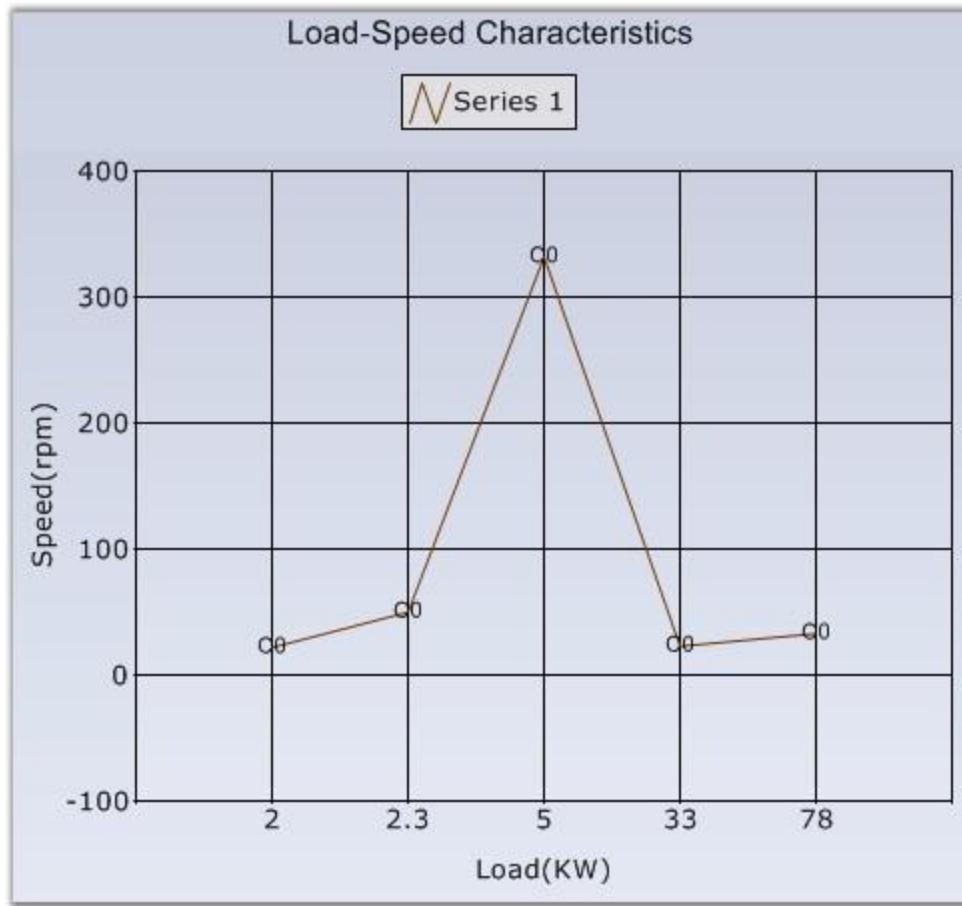


Figure 246: Non-Indexed X Values



*Figure 247 Indexed X Values*



**Note:** Indexing is supported only on the x-axis in Essential Chart.

You can enable x-axis indexing or categorizing through the **Indexed** property of the ChartControl as shown below:

[C#]

```
this.chartControl1.Indexed = true;
```

[VB .NET]

```
Me.chartControl1.Indexed = True
```

The above property automatically affects all the x-axes in the chart.

You can also optionally customize the labels of the points in such an indexed series as explained in [Chart Labels Customization](#).

## 4.6.2 Inverted Axis

Essential Chart provides support for inverting the values in an axis. Data on an inverted axis is plotted in the opposite direction - top to bottom for y-axis and right to left for x-axis. To enable this behavior, set the **ChartAxis.Inversed** to **true**.

Chart Axis Property	Description
Inversed	Indicates whether the axis should be reversed.

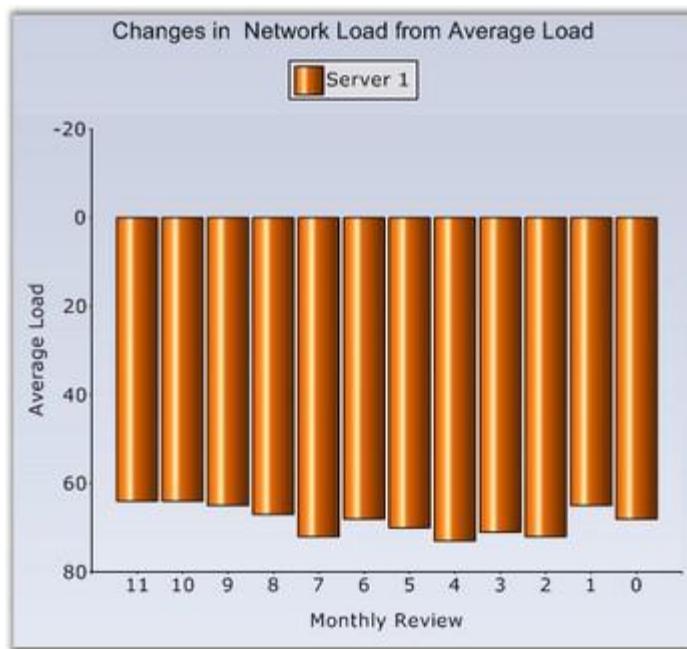
### [C#]

```
// This inverts the specified chart axis.  
this.chartControl1.PrimaryXAxis.Inversed = true;  
this.chartControl1.PrimaryYAxis.Inversed = true;
```

### [VB .NET]

```
' This inverts the specified chart axis.  
Me.chartControl1.PrimaryXAxis.Inversed = True  
Me.chartControl1.PrimaryYAxis.Inversed = True
```

The following image shows a chart whose x and y axes have been reversed.



*Figure 248: Chart with both Axes Reversed*

### 4.6.3 Opposed Axis

For every chart type there is an implied x-axis and y-axis position and by default all the x-axes and y-axes will be rendered in that corresponding position. You can override this default behavior by setting the **OpposedPosition** property to **true** for an axis which will cause it to be rendered in a side opposite to that of the implied position.

#### [C#]

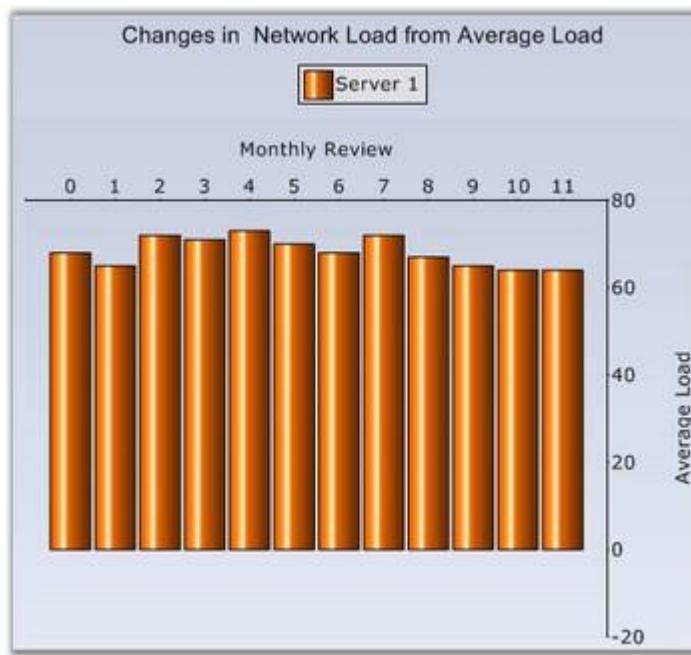
```
// Will cause the X axis to be rendered on top instead of the default
// bottom position
this.chartControl1.PrimaryXAxis.OpposedPosition = true;
// Will cause the Y axis to be rendered on the right instead of the
// default left position
this.chartControl1.PrimaryYAxis.OpposedPosition = true;
```

#### [VB .NET]

```
' Will cause the X axis to be rendered on top instead of the default
' bottom position
Me.chartControl1.PrimaryXAxis.OpposedPosition = True
' Will cause the Y axis to be rendered on the right instead of the
```

```
default left position  
Me.chartControll.PrimaryYAxis.OpposedPosition = True
```

The above code snippet will place both the x and y-axes in the position opposite to their default implied position.



*Figure 249: Chart displaying Opposed X and Y Axes*

You can similarly set this property on any custom **ChartAxis** that you might add to the chart. Using multiple axes in a chart is described in this topic: [Multiple Axes](#).

The OpposedPosition along with Inversed setting could be useful for implementing charts for right-to-left cultures.

#### 4.6.4 Multiple Axes

Often you will have to plot multiple series on a single chart, each in its own x or y axis. You will then need to add an x or y axis to the chart in addition to the already existing PrimaryXAxis and PrimaryYAxis. You can do this by instantiating a **ChartAxis** and adding it to the **Axes** collection. Then specify the newly created axis as the x or y axis of a particular series.

The following are the steps to include a new axis to the chart.

[C#]

```
// Create a new instance of the chart axis.  
private ChartAxis secXAxis = new ChartAxis();  
  
// Add the secondary axis to the chart axis collection.  
this.chartControl1.Axes.Add(this.secXAxis);  
  
// Specify this axis to be the axis for an existing series  
this.chartControl1.Series[1].XAxis = this.secXAxis;
```

[VB.NET]

```
' Create a new instance of the chart axis.  
Private secXAxis As ChartAxis = New ChartAxis()  
  
' Add the secondary axis to the chart axis collection.  
Me.chartControl1.Axes.Add(Me.secXAxis)  
  
' Specify this axis to be the axis for an existing series  
Me.chartControl1.Series(1).XAxis = Me.secXAxis
```

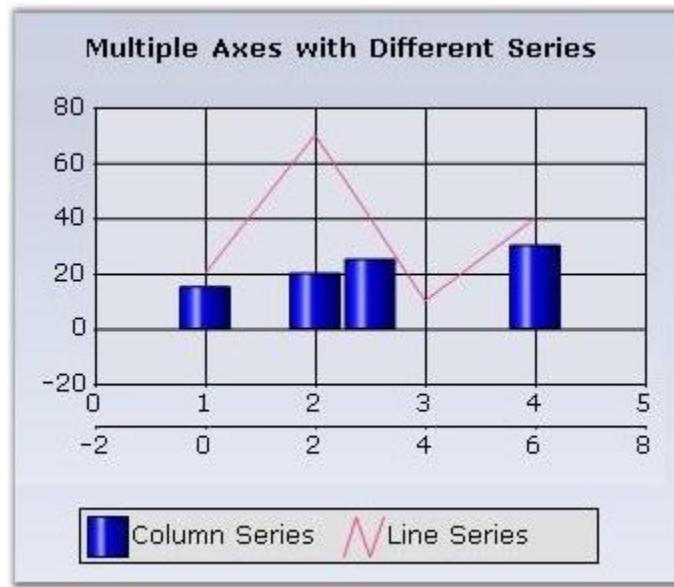
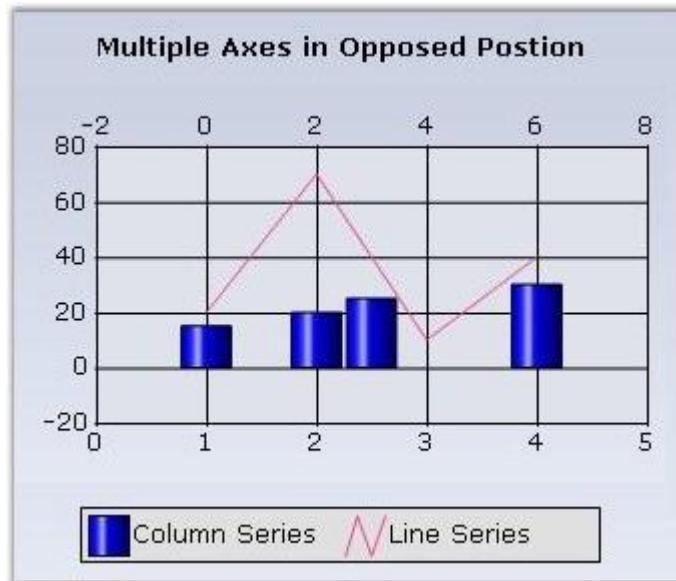


Figure 250: ChartControl with a 2nd X-Axis (-2 to 8) stacked below the Primary X-Axis

**Opposed Position**

By default, this additional axis will be rendered right next to the corresponding primary axis as seen above. This might be undesirable and you would instead want it to be rendered at the opposite side of the primary axis. This is done by setting the **OpposedPosition** property to **true**. Read more on Opposed Axis here.



*Figure 251: ChartControl with a 2nd X-Axis in Opposed Position*

### **Stacked or SideBySide Position**

By default, the secondary axes are rendered stacked over, or parallel, to the corresponding primary axis. And also sometimes rendered in a position opposite to the primary axis as shown in the above screenshots. This is because the **XAxisLayoutMode** and **YAxisLayoutMode** properties are set to **Stacking** by default.

However, you might want the secondary axis to be rendered in-line, side-by-side to the primary axis. You can do by setting the **XAxisLayoutMode** and **YAxisLayoutMode** properties to **SideBySide**.

Here is a code sample.

[C#]

```
this.ChartControl1.ChartArea.XAxesLayoutMode =  
ChartAxesLayoutMode.SideBySide;
```

[VB.NET]

```
Me.ChartControll.ChartArea.XAxesLayoutMode =  
ChartAxesLayoutMode.SideBySide;
```

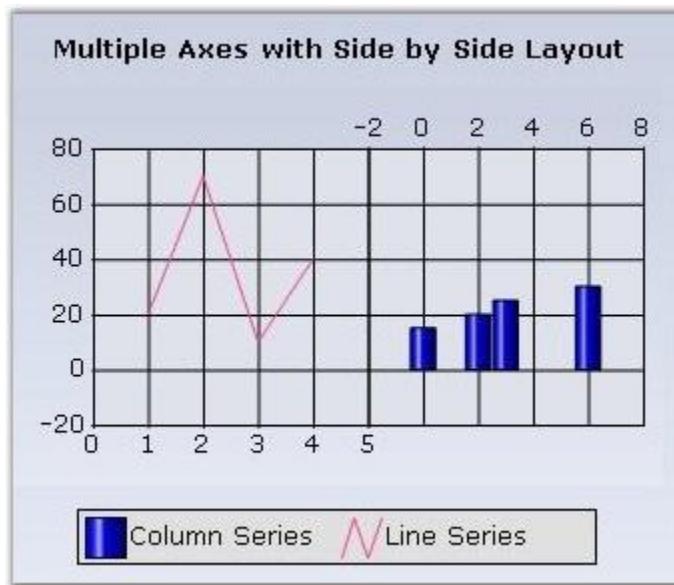


Figure 252: ChartControl with SideBySide Layout of Multiple Axes

### ChartAxesLayouts

You can now combine the stacking and side-by-side chart axes layouts when multiple Axes are used, as shown in the below image. Using this feature, it is possible to position the three Y axis, as one on right side and the second one on the same side and third one on the opposite side.



*Figure 253: Combining Stacking and SideBySide Layout*

[C#]

```
//Created chart axes:  
  
ChartAxis axis = this.chartControl1.PrimaryYAxis;  
ChartAxis axis0 = new ChartAxis(ChartOrientation.Vertical);  
ChartAxis axis1 = new ChartAxis(ChartOrientation.Vertical);  
  
//Added chart axes into the chart:  
  
chartControl1.Axes.Add(axis0);  
chartControl1.Axes.Add(axis1);  
  
//Created chart axis layout using ChartAxisLayout class: (New class)  
  
ChartAxisLayout layout1 = new ChartAxisLayout();  
  
//Added the axes to this layout including the primary axis:
```

```
layout1.Axes.Add(axis);
layout1.Axes.Add(axis0);
layout1.Axes.Add(axis1);

//Added the layout into ChartArea.

chartControl1.ChartArea.YLayouts.Add(layout1);
```

**[VB.NET]**

```
'Created chart axes:

Dim axis As ChartAxis = Me.chartControl1.PrimaryYAxis
Dim axis0 As New ChartAxis(ChartOrientation.Vertical)
Dim axis1 As New ChartAxis(ChartOrientation.Vertical)

'Added chart axes into the chart:

chartControl1.Axes.Add(axis0)
chartControl1.Axes.Add(axis1)

'Created chart axis layout using ChartAxisLayout class:(New class)

Dim layout1 As New ChartAxisLayout()

'Added the axes to this layout including the primary axis:

layout1.Axes.Add(axis)
layout1.Axes.Add(axis0)
layout1.Axes.Add(axis1)

'Added the layout into ChartArea.

chartControl1.ChartArea.YLayouts.Add(layout1)
```



**Note:** All the axes with the same orientation must be added to ChartAxisLayout (PrimaryAxis as well) as illustrated in the above code snippet.

#### 4.6.5 Axis Value Type

You can set the value type for an axis using **Axes.ValueType** property. You can set any of the following value types of which the default is **double**.

- double
- datetime
- logarithmic

If you set the **ValueType** to 'Logarithmic', then you need to specify the log base for the axis using **Axes.LogBase** property. The default value of LogBase is 10.

[C#]

```
this.chartControl1.PrimaryXAxis.ValueType = ChartValueType.Logarithmic;
this.chartControl1.PrimaryXAxis.LogBase = 3;
```

## See Also

[Axis Range and Intervals](#)

## 4.6.6 Axis Range and Intervals

### Automatic Range Calculation

The range and intervals for an axis are automatically calculated by the built-in "nice range calculation engine", by default. This engine takes a raw data series and comes up with a nice human readable range of numbers in which to represent them. For example, if the data series contains points in the range 1.2 - 3.7, the engine would come up with a scale of 0 - 5 for the axis with 10 intervals of 0.5 each.

This default behavior is controlled by the **ChartAxis.RangeType** property which is set to **Auto** by default.

### Specifying Custom Ranges

Sometimes the automatic range generation might not be good enough for you, in which case you can specify a custom range on the axis. You should start by setting the **ChartAxis.RangeType** property to **Set**. Then use one of the following properties to specify a custom range.

ChartAxis Property	Applies to RangeType	Applies to ValueType	Description
Range	Set	Double	Specifies the minimum, maximum and interval for the axis. Use this if the data points are of double type.

DateTimeRange	Set	DateTime	Specifies the start and end dates and interval time for the axis. Use this if the data points are of datetime type.
---------------	-----	----------	---------------------------------------------------------------------------------------------------------------------

Here is some sample code that shows how this is done.

#### [C#]

```
// Customize the X axis range and interval which has points of type
// DateTime
this.chartControl1.PrimaryXAxis.RangeType = ChartAxisRangeType.Set;
this.chartControl1.PrimaryXAxis.ValueType = ChartValueType.DateTime;
this.chartControl1.PrimaryXAxis.DateTimeRange = new
ChartDateTimeRange(baseDate.AddMonths(-1), baseDate.AddMonths(6), 1,
ChartDateTimeIntervalType.Months);

// Customize the Y axis range and interval which has points of type
double
this.chartControl1.PrimaryYAxis.RangeType = ChartAxisRangeType.Set;
this.chartControl1.PrimaryYAxis.Range = new MinMaxInfo(1, 20, 2);

// Customize the Y axis range and interval which has points of type
double
this.chartControl1.PrimaryXAxis.RangeType = ChartAxisRangeType.Set;
this.chartControl1.PrimaryXAxis.Range = new MinMaxInfo(0, 6, 1);
```

#### [VB.NET]

```
' Customize the X axis range and interval which has points of type
// DateTime
Me.chartControl1.PrimaryXAxis.RangeType = ChartAxisRangeType.Set
Me.chartControl1.PrimaryXAxis.ValueType = ChartValueType.DateTime
Me.chartControl1.PrimaryXAxis.DateTimeRange = New
ChartDateTimeRange(baseDate.AddMonths(-1), baseDate.AddMonths(6), 1,
ChartDateTimeIntervalType.Months)

' Customize the Y axis range and interval which has points of type
double
Me.chartControl1.PrimaryYAxis.RangeType = ChartAxisRangeType.Set
Me.chartControl1.PrimaryYAxis.Range = New MinMaxInfo(1, 20, 2)

' Customize the x axis range and interval which has points of type
double
Me.chartControl1.PrimaryXAxis.RangeType = ChartAxisRangeType.Set
```

```
Me.chartControll.PrimaryXAxis.Range = New MinMaxInfo(0, 6, 1)
```

You can however tweak the ranges and intervals that get generated through these properties.

### Changing Intervals

Use these properties to customize the intervals that get generated:

ChartAxis Property	Applies to RangeType	Applies to ValueType	Description
DesiredIntervals	Auto	Double, DateTime	A request for the nice-range calculation engine to come up with a nice range with so many intervals. The engine will only use this setting as a guidance. Default value is 6.
IntervalType	Auto	DateTime	Specifies whether the interval that gets calculated should be in Years, Months, Weeks, Days, Hours, Minutes, Seconds or Milliseconds. This setting is used only if the <b>ValueType</b> of the axis is set to <b>DateTime</b> . Default value is <b>Auto</b> .

### Changing Origin

Use these properties to customize the origin of the axes:

ChartAxis Property	Applies to RangeType	Applies to ValueType	Description
PreferZero	Auto	Double	Indicates whether one boundary of the calculated range should be tweaked to zero. Such tweaking will happen only if zero is within a reasonable distance from the calculated boundary. To ensure that one boundary is always zero, use the

			"ForceZero" setting instead. Default value is <b>true</b> .
ForceZero	Auto	Double	Indicates whether one boundary of the calculated range should always be tweaked to zero. Default value is <b>true</b> .
CustomOrigin	Auto and Set	Double, DateTime	Lets you use the properties Origin and OriginDate below. Default value is <b>false</b> .
Origin	Auto and Set	Double	Lets you specify a custom origin (double value) for the axis. Use this property when the data points are of double type. The interval and range will then be calculated automatically. Remember to set CustomOrigin to <b>true</b> . Default value is <b>0.0</b> .
OriginDate	Auto and Set	DateTime	Lets you specify a custom origin (double value) for the axis. Use this property when the data points are of double type. The interval and range will then be calculated automatically. Remember to set CustomOrigin to <b>true</b> . Default value is <b>DateTime.MinValue</b> .
Offset	Auto and Set	Double and DateTime	Specifies the offset that should be applied to the automatically calculated start of the range.
OffsetDateTime	Auto	DateTime	Specifies the offset that should be applied to the automatically calculated start of the range.
DateTimeInterval.Offs	Set	DateTime	Use this instead of Offset if

OffsetType			you want to specify the OffsetType (see below).
DateTimeInterval.OffsetType	Set	DateTime	Specifies the type of offset specified above. Could be Auto, Years, Months, Weeks, Days, Hours, Minutes, Seconds or Milliseconds.
RangePaddingType	Auto	Double and DateTime	Specifies if there should be any padding applied between the points and the axes, before and after the datapoints.

#### 4.6.7 Axis Dimensions

The axis' starting point, length and the whole rectangle (comprising the axis and it's labels) can be customized using the following properties.

ChartAxis Property	Description
Location	Specifies the starting location of the axis. <b>LocationType</b> property should be equal to <b>Set</b> to set the Location property.
LocationType	<ul style="list-style-type: none"> <li>• <b>Set</b> - To be able to use the above Location property.</li> <li>• <b>Auto</b> - Axis position will be automatically calculated to prevent overlap with the labels. (Default value)</li> <li>• <b>AntiLabelCut</b> - Axis thickness is calculated and the corresponding axis will be placed automatically, to prevent cutting of the labels by the sides of the control. Doing this preserves one co-ordinate of the axis location (X coordinate for horizontal axis and y coordinate for vertical axis).</li> </ul>
AutoSize	Specifies whether length of an axis is calculated automatically or specified via the Size property.
Size	Lets you specify the length of the axis. Uses the x value for x-axis and y-value for y-axis. Increasing or decreasing the default length will cause the intervals to expand or shrink correspondingly. The AutoSize should be set to <b>false</b> for this property to be used.

Rect	Specifies the rectangle that includes the axis and its labels. This provides great flexibility in letting you customize the position and size of the axes.
------	------------------------------------------------------------------------------------------------------------------------------------------------------------

### Illustrating Custom Axis Location

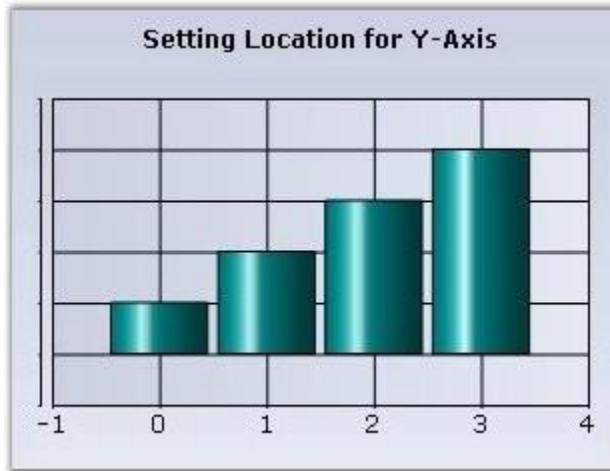


Figure 254: Chart Axis with Location Properties Set

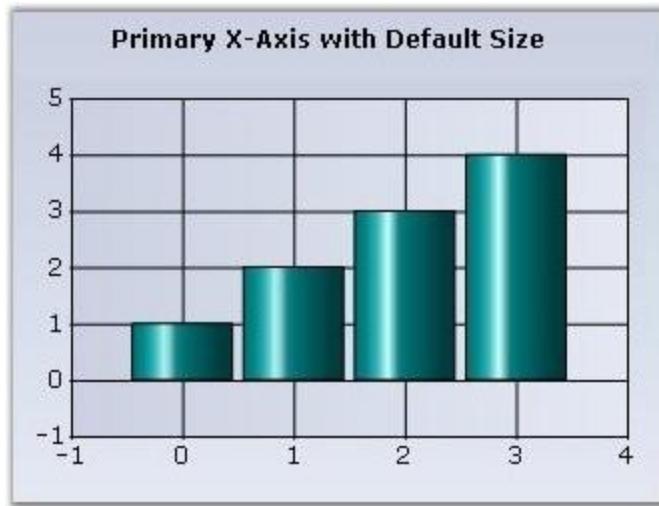
#### [C#]

```
this.chartControl1.PrimaryYAxis.LocationType =
Syncfusion.Windows.Forms.Chart.ChartAxisLocationType.Set;
this.chartControl1.PrimaryYAxis.Location = new PointF(15, 200);
```

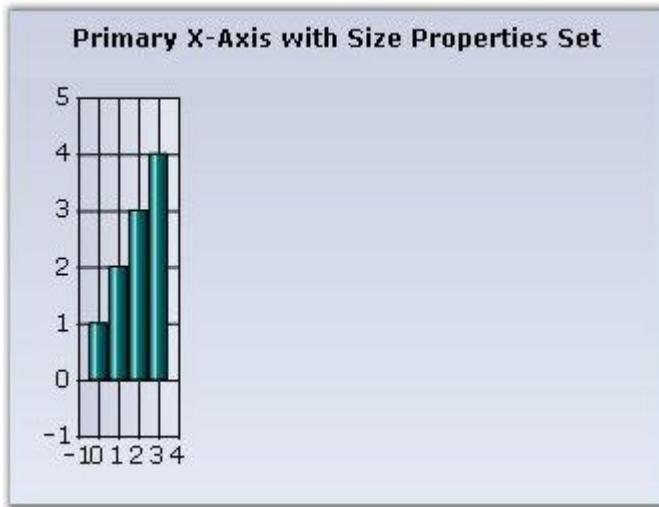
#### [VB]

```
Me.ChartControl1.PrimaryYAxis.LocationType =
Syncfusion.Windows.Forms.Chart.ChartAxisLocationType.Set
Me.ChartControl1.PrimaryYAxis.Location = New PointF(15, 200)
```

### Illustrating Custom Axis Size



*Figure 255: Chart rendered in AutoSize Mode*



*Figure 256: Chart rendered with a custom size for the Primary X-Axis*

[C#]

```
this.chartControl1.PrimaryXAxis.AutoSize = false;  
this.chartControl1.PrimaryXAxis.Size = new Size(50, 20);
```

[VB]

```
Me.ChartControl1.PrimaryXAxis.AutoSize = False  
Me.ChartControl1.PrimaryXAxis.Size = New Size(50, 20)
```

## 4.6.8 Axis Labels

This section talks about the customization of axis labels in the following topics:

### 4.6.8.1 Axis Label Text Formatting, Appearance and Positioning

By default, the label texts are automatically determined based on the axis data points and the generated intervals. You can better the format, look and positioning of the labels using the properties listed below.

ChartAxis Property	Description
Format	If the data points are double values then use this property to specify the format in which to render the double value. The specified format will be used in the Double.ToString method to arrive at the formatted string. Search MSDN documentation for <b>Standard Numeric Format Strings</b> for more information on the format strings.
DateTimeFormat	If the data points are <b>DateTime</b> values, then use this property to specify the format in which to render the date. The specified format will be used in the DateTime.ToString() method to arrive at the formatted string. Search MSDN documentation for <b>Date and Time Format Strings</b> for more information on the format strings.
ForeColor	Affects the labels and other text colors that gets rendered in the axis.
Font	Specifies the font to use for label and other texts that get rendered in the axis. By default it is set to Trebuchet, 9, regular.
ScaleLabels	Setting this to <b>true</b> will automatically resize the text if the chart size is expanded by the user.
LabelAlignment	Specifies if the label should be rendered Near, Far or Center within the available area. Default is <b>Center</b> .
LabelRotate	Specifies whether or not labels should be rotated. Use the <b>LabelRotateAngle</b> to specify the angle.

LabelRotateAngle	If <b>LabelRotate</b> is <b>true</b> , this property specifies the angle of rotation.
------------------	---------------------------------------------------------------------------------------

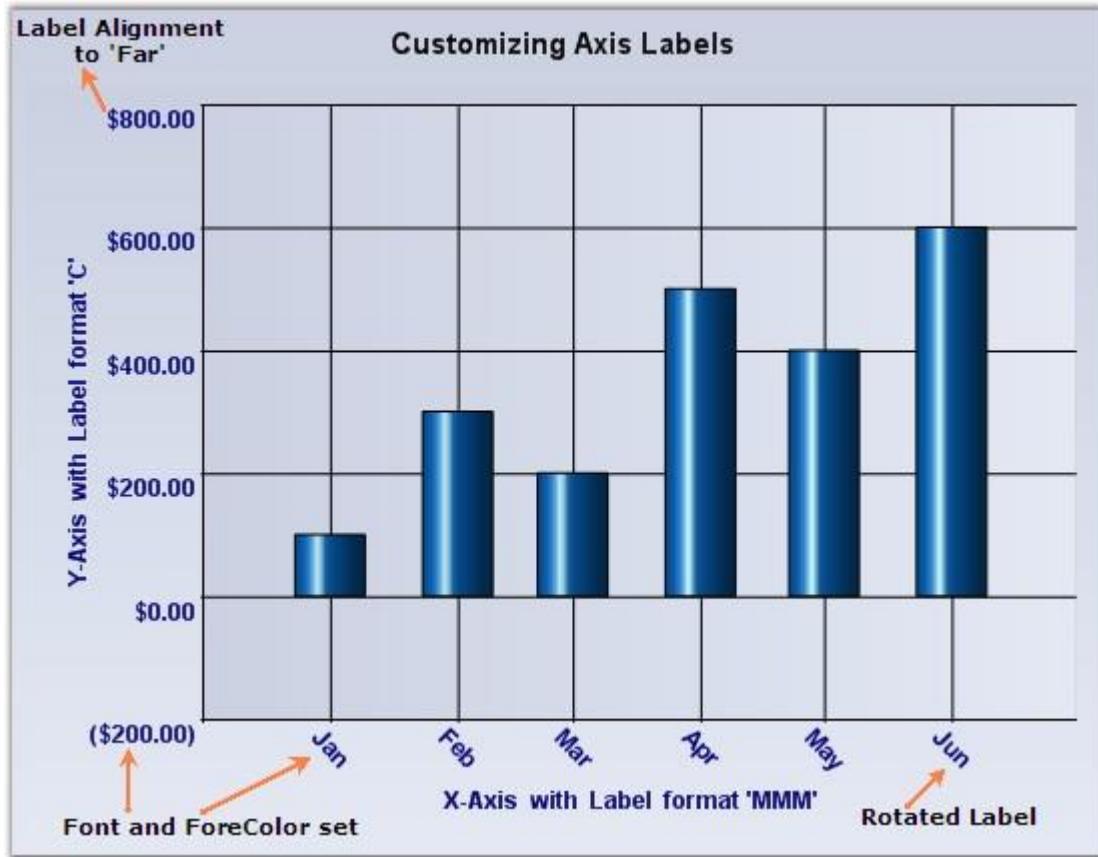


Figure 257: Axis Label Customization

[C#]

```
//Settings datetime format to Xaxis
this.chartControl1.PrimaryXAxis.DateTimeFormat = "MMM";

//Settings format to Yaxis
this.chartControl1.PrimaryYAxis.ValueType = ChartValueType.Double;
this.chartControl1.PrimaryYAxis.Format = "C";

//setting ForeColor and Font to axes labels
this.chartControl1.PrimaryXAxis.ForeColor = System.Drawing.Color.Navy;
this.chartControl1.PrimaryXAxis.Font = new System.Drawing.Font("Arial",
9F, System.Drawing.FontStyle.Bold);
this.chartControl1.PrimaryYAxis.ForeColor = System.Drawing.Color.Navy;
```

```
this.chartControll1.PrimaryYAxis.Font = new System.Drawing.Font("Arial",  
9F, System.Drawing.FontStyle.Bold);  
  
//Label property settings for X-Axis  
this.chartControll1.PrimaryXAxis.LabelAligment =  
System.Drawing.StringAlignment.Center;  
this.chartControll1.PrimaryXAxis.LabelRotate = true;  
this.chartControll1.PrimaryXAxis.LabelRotateAngle = 45;  
  
//Label property settings for Y-Axis  
this.chartControll1.PrimaryYAxis.LabelAligment =  
System.Drawing.StringAlignment.Far;
```

**[VB]**

```
'Settings datetime format to Xaxis  
Me.chartControll1.PrimaryXAxis.DateTimeFormat = "MMM"  
  
Settings format to Yaxis  
Me.chartControll1.PrimaryYAxis.ValueType = ChartValueType.Double  
Me.chartControll1.PrimaryYAxis.Format = "D"  
  
'Setting ForeColor and Font to axes labels  
Me.chartControll1.PrimaryXAxis.ForeColor = System.Drawing.Color.Navy  
Me.chartControll1.PrimaryXAxis.Font = new System.Drawing.Font("Arial",  
9F, System.Drawing.FontStyle.Bold)  
Me.chartControll1.PrimaryYAxis.ForeColor = System.Drawing.Color.Navy  
Me.chartControll1.PrimaryYAxis.Font = new System.Drawing.Font("Arial",  
9F, System.Drawing.FontStyle.Bold)  
  
'Label property settings for X-Axis  
Me.chartControll1.PrimaryXAxis.LabelAligment =  
System.Drawing.StringAlignment.Center  
Me.chartControll1.PrimaryXAxis.LabelRotate = true  
Me.chartControll1.PrimaryXAxis.LabelRotateAngle = 45  
  
'Label property settings for Y-Axis  
Me.chartControll1.PrimaryYAxis.LabelAligment =  
System.Drawing.StringAlignment.Far
```

#### 4.6.8.2 Customizing Label Text

The formatting options above will usually satisfy the label text requirements. However, there are many other scenarios where this might not be sufficient. Here are some ways to customize the text rendered in the label.

### Customizing the label text for the automatically generated intervals

ChartAxis Event	Description
ChartFormatAxisLabel	The event that gets raised for each label before getting rendered. This is a good place to customize the label text.

The following ChartFormatAxisLabelEventArgs properties provide information specific to this event.

ChartFormatAxisLabel Property	Description
AxisOrientation	Returns the orientation of the axis for which the label is being generated.
Handled	Indicates whether this event was handled and no further processing is required from the chart.
IsAxisPrimary	Indicates whether the axis for which the label is being generated is a primary axis.
Label	Gets / sets the label that is to be rendered.
Value	Returns the value associated with the position of the label.
ValueAsDate	Returns the value associated with the position of the label as <b>DateTime</b> .
Tooltip	Specifies the content of the tooltip.

[C#]

```
private void chartControl1_ChartFormatAxisLabel(object sender,
ChartFormatAxisLabelEventArgs e)
{
    if (e.AxisOrientation == ChartOrientation.Horizontal)
    {
        if (e.ValueAsDate.Month == 1)
            e.Label = "1st Month";
        else if (e.ValueAsDate.Month == 2)
            e.Label = "2nd Month";
        else if (e.ValueAsDate.Month == 3)
            e.Label = "3rd Month";
        else if (e.ValueAsDate.Month == 4)
            e.Label = "4th Month";
        else if (e.ValueAsDate.Month == 5)
            e.Label = "5th Month";
        else if (e.ValueAsDate.Month == 6)
            e.Label = "6th Month";
        e.Handled = true;
    }
}
```

[VB]

```
Private Sub chartControl1_ChartFormatAxisLabel(ByVal sender As Object,
ByVal e As ChartFormatAxisLabelEventArgs)
    If e.AxisOrientation = ChartOrientation.Horizontal Then
        If e.ValueAsDate.Month = 1 Then
            e.Label = "1st Month"
        ElseIf e.ValueAsDate.Month = 2 Then
            e.Label = "2nd Month"
        ElseIf e.ValueAsDate.Month = 3 Then
            e.Label = "3rd Month"
        ElseIf e.ValueAsDate.Month = 4 Then
            e.Label = "4th Month"
        ElseIf e.ValueAsDate.Month = 5 Then
            e.Label = "5th Month"
        ElseIf e.ValueAsDate.Month = 6 Then
            e.Label = "6th Month"
        End If
        e.Handled = True
    End If
End Sub
```

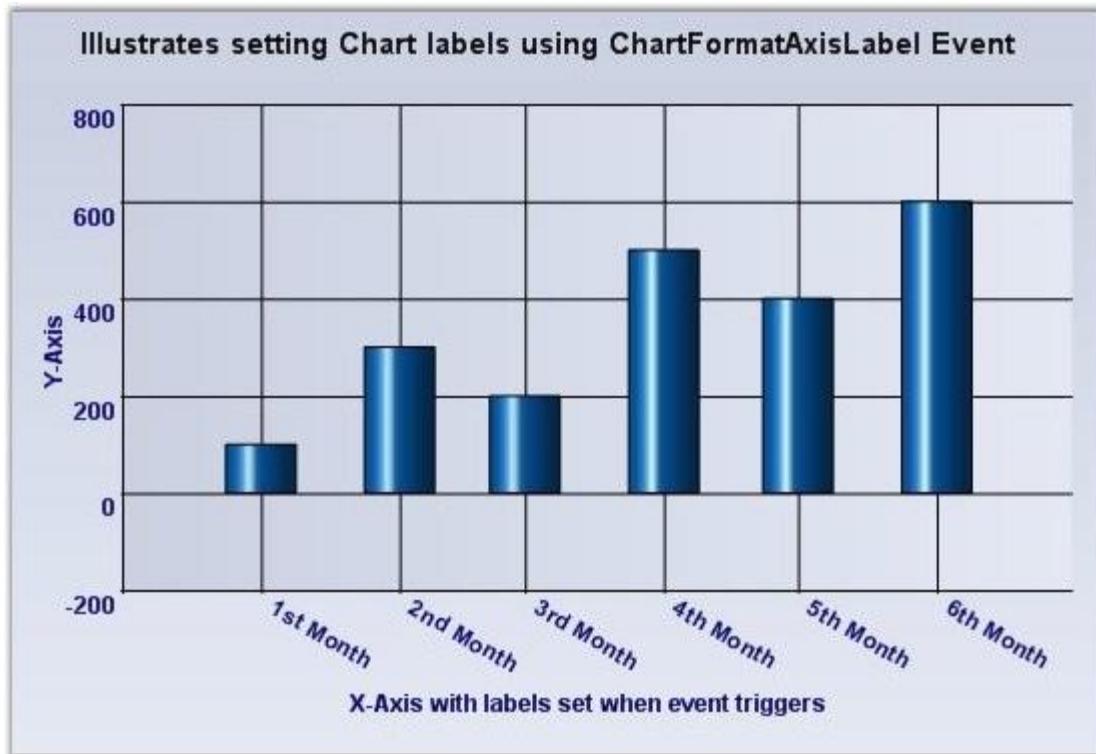


Figure 258: Customized Chart Labels

Specify a set of custom labels thereby dictating the intervals as well

#### 1. Using Custom Text

ChartAxis Property	Description
TickLabelsDrawingMode	<p><b>AutomaticMode</b> - Labels will be determined by the engine.</p> <p><b>UserMode</b> - Labels from the Labels collection will be used.</p> <p><b>BothUserAndAutomaticMode</b> - Both labels from the automatic mode and user mode will be rendered.</p> <p><b>None</b> - Labels will not be rendered.</p>
Labels	A custom collection that lets you fully customize the labels that gets generated. The TickLabelsDrawingMode should be set to <b>UserMode</b> or <b>BothUserAndAutomaticMode</b> .

**[C#]**

```
//Setting drawing mode
this.chartControll1.PrimaryXAxis.TickLabelsDrawingMode =
ChartAxisTickLabelDrawingMode.UserMode;

//Adding new labels
this.chartControll1.PrimaryXAxis.Labels.Add(new ChartAxisLabel("Q1 Mid
Point", Color.OrangeRed, new Font("Arial", 8F,
System.Drawing.FontStyle.Bold), new DateTime(2007, 2, 15), "", "",
ChartValueType.Custom));
this.chartControll1.PrimaryXAxis.Labels.Add(new ChartAxisLabel("Q2 Mid
Point", Color.OrangeRed, new Font("Arial", 8F,
System.Drawing.FontStyle.Bold), new DateTime(2007, 5, 15), "", "",
ChartValueType.Custom));
```

**[VB]**

```
'Setting drawing mode
Me.chartControll1.PrimaryXAxis.TickLabelsDrawingMode =
ChartAxisTickLabelDrawingMode.UserMode

'Adding new labels
Me.chartControll1.PrimaryXAxis.Labels.Add(New ChartAxisLabel("Q1 Mid
Point", Color.OrangeRed, New Font("Arial", 8F,
System.Drawing.FontStyle.Bold), New DateTime(2007, 2, 15), "", ""),
ChartValueType.Custom))
Me.chartControll1.PrimaryXAxis.Labels.Add(New ChartAxisLabel("Q2 Mid
Point", Color.OrangeRed, New Font("Arial", 8F,
System.Drawing.FontStyle.Bold), New DateTime(2007, 5, 15), "", ""),
ChartValueType.Custom))
```

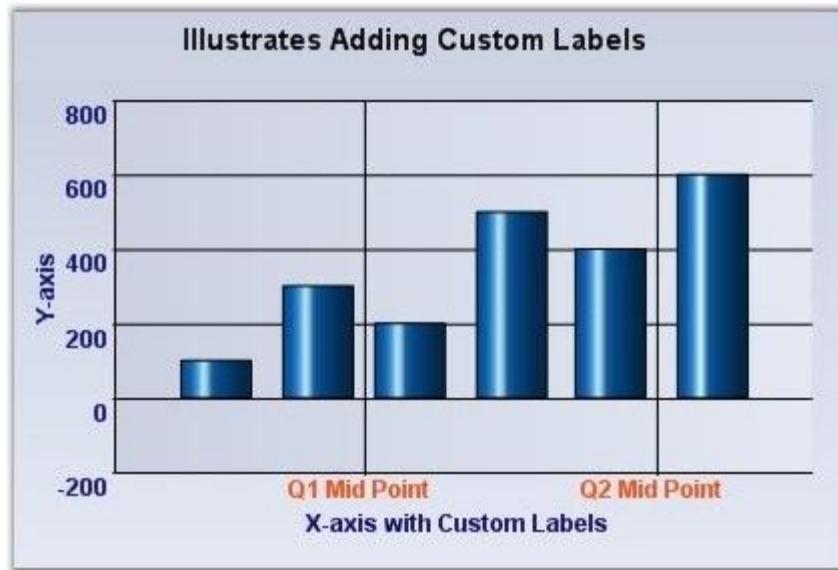


Figure 259: Chart with 'Q1 Mid Point' and 'Q2 Mid Point' Custom Labels

## 2. Using Formatted Text

[C#]

```
//Setting drawing mode
this.chartControl1.PrimaryXAxis.TickLabelsDrawingMode =
ChartAxisTickLabelDrawingMode.UserMode;

'Adding new labels
this.chartControl1.PrimaryXAxis.Labels.Add(new ChartAxisLabel("", 
Color.Maroon, new Font("Arial", 9F, System.Drawing.FontStyle.Bold), new 
DateTime(2007, 2, 15), "", "M", ChartValueType.DateTime));
this.chartControl1.PrimaryXAxis.Labels.Add(new ChartAxisLabel("", 
Color.Maroon, new Font("Arial", 9F, System.Drawing.FontStyle.Bold), new 
DateTime(2007, 05, 15), "", "M", ChartValueType.DateTime));
```

[VB]

```
'Setting drawing mode
Me.chartControl1.PrimaryXAxis.TickLabelsDrawingMode =
ChartAxisTickLabelDrawingMode.UserMode
'Clearing all the default labels
this.chartControl1.PrimaryXAxis.Labels.Clear()
'Adding new labels
Me.chartControl1.PrimaryXAxis.Labels.Add(New ChartAxisLabel("", 
Color.Maroon, New Font("Arial", 9F, System.Drawing.FontStyle.Bold), New 
DateTime(2007, 2, 15), "", "M", ChartValueType.DateTime))
```

```
Me.chartControll.PrimaryXAxis.Labels.Add(New ChartAxisLabel("", Color.Maroon, New Font("Arial", 9F, System.Drawing.FontStyle.Bold), New DateTime(2007, 05, 15), "", "M", ChartValueType.DateTime))
```

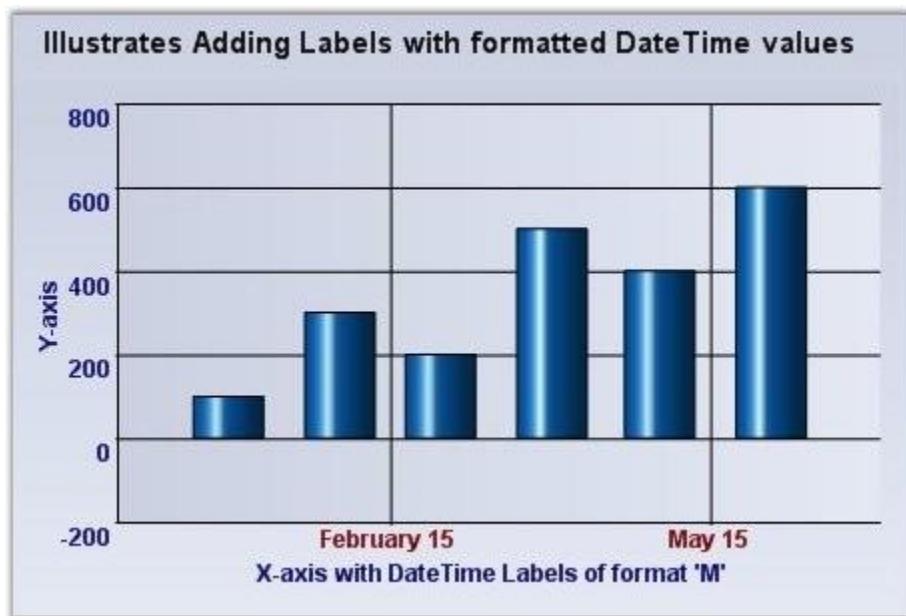


Figure 260: DateTime formatted labels at the specified Intervals

#### 4.6.8.3 Intersecting Labels

Sometimes the chart dimensions could cause the labels to intersect. The chart will, by default, render those texts one over the other. But, it also has some built-in capabilities to workaround this overlap and lets you dictate the technique to follow. Refer to the properties below.

ChartAxis Property	Description
LabelIntersectAction	Specifies the action to take when labels texts intersect. <ul style="list-style-type: none"> <li>• <b>MultipleRows</b> - Will render the labels in multiple rows.</li> <li>• <b>None</b> - Do nothing (default value)</li> <li>• <b>Rotate</b> - Rotates text so as to avoid overlap</li> <li>• <b>Wrap</b> - wraps text.</li> </ul>
EdgeLabelsDrawingMode	Affects the labels that get rendered at the edges of the axis. Possible values: <ul style="list-style-type: none"> <li>• <b>Center</b> - Centers the label at the interval. Default setting.</li> <li>• <b>Shift</b> - Shifts the labels so that it's within the interval boundaries</li> </ul>

	<ul style="list-style-type: none"> <li>• <b>ClippingProtection</b> - Uses some intelligent logic to avoid clipping.</li> </ul>
HidePartialLabels	When this property is set to true and when label overlap occurs, the chart will selectively hide certain labels (usually the min and max labels to begin with) to keep the rest of labels readable. Default value is false.

**[C#]**

```
this.ChartWebControl1.PrimaryXAxis.HidePartialLabels = true;
this.ChartWebControl1.PrimaryXAxis.LabelIntersectAction =
ChartLabelIntersectAction.Rotate;
```

**[VB]**

```
Me.ChartWebControl1.PrimaryXAxis.HidePartialLabels = True
Me.ChartWebControl1.PrimaryXAxis.LabelIntersectAction =
ChartLabelIntersectAction.Rotate
```

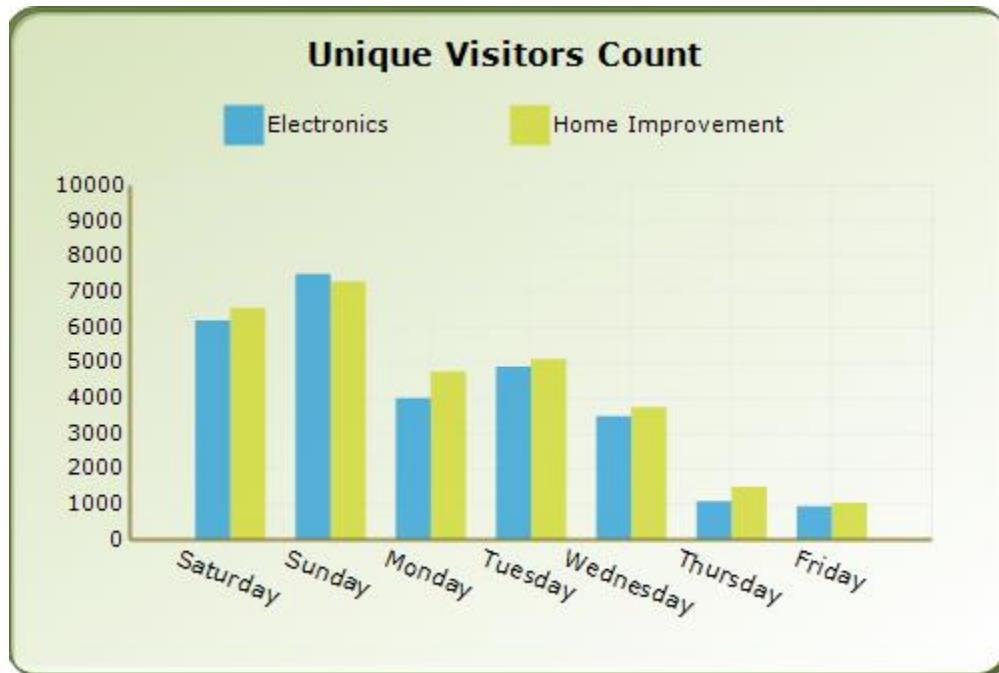


Figure 261: Intersecting Labels

#### 4.6.8.4 Grouping Labels

Another interesting feature that is available is to be able to group a set of adjoining labels and mark them with a new label. For example, grouping the first three months of the year and marking them as Q1 and so on. The following properties will let you do that.

ChartAxis Property	Description
GroupingLabels	Lets you group a range of default labels and provide them a custom name/label.
DrawTickLabelGrid	Puts the labels within a grid. Though commonly used when in grouping mode, this feature can be used even otherwise.

**[C#]**

```
ChartAxisGroupingLabel Q1 = new ChartAxisGroupingLabel(new DoubleRange(1, 3), "Q1");
Q1.BorderStyle = ChartAxisGroupingLabelBorderStyle.Rectangle;
Q1.Font = new Font("Arial", 10F, FontStyle.Bold);
this.chartControl1.PrimaryXAxis.GroupingLabels.Add(Q1);

ChartAxisGroupingLabel Q2 = new ChartAxisGroupingLabel(new DoubleRange(4, 6), "Q2");
Q2.BorderStyle = ChartAxisGroupingLabelBorderStyle.Rectangle;
Q2.Font = new Font("Arial", 10F, FontStyle.Bold);
this.chartControl1.PrimaryXAxis.GroupingLabels.Add(Q2);

this.chartControl1.PrimaryXAxis.DrawTickLabelGrid = true;
```

**[VB]**

```
Dim Q1 As New ChartAxisGroupingLabel(New DoubleRange(1, 3), "Q1")
Q1.BorderStyle = ChartAxisGroupingLabelBorderStyle.Rectangle
Q1.Font = New Font("Arial", 10.0F, FontStyle.Bold)
Me.chartControl1.PrimaryXAxis.GroupingLabels.Add(Q1)

Dim Q2 As New ChartAxisGroupingLabel(New DoubleRange(4, 6), "Q2")
Q2.BorderStyle = ChartAxisGroupingLabelBorderStyle.Rectangle
Q2.Font = New Font("Arial", 10.0F, FontStyle.Bold)
Me.chartControl1.PrimaryXAxis.GroupingLabels.Add(Q2)

Me.chartControl1.PrimaryXAxis.DrawTickLabelGrid = True
```

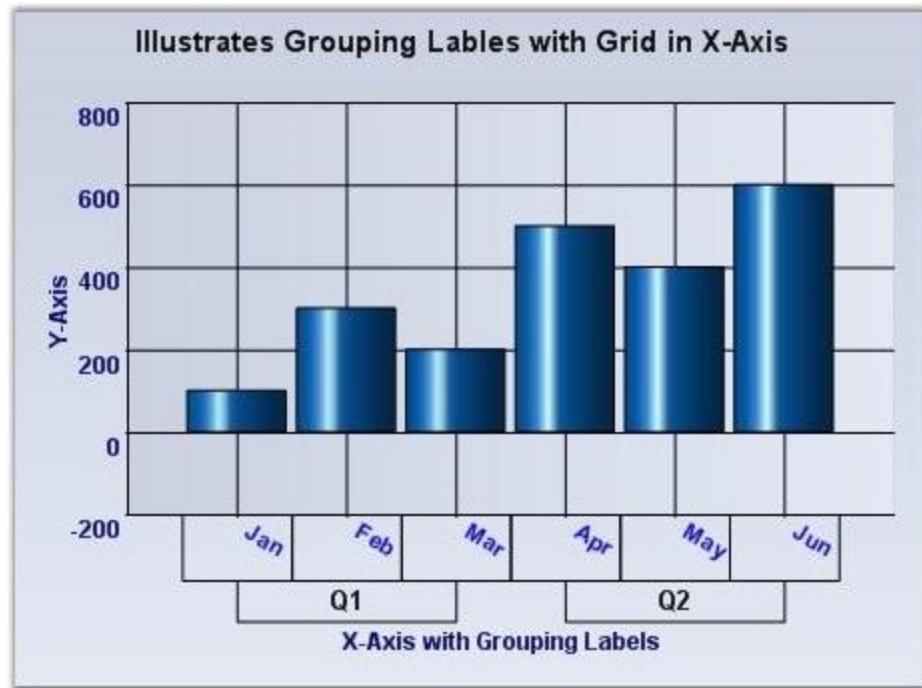


Figure 262: Q1 and Q2 Grouping Labels with Grids

#### 4.6.8.5 Tooltip Support for ChartAxisLabels

Essential Chart provides tooltip support for ChartAxisLabel. By default ChartAxisLabel will be displayed as tooltip. You can also customize the tooltip to show any content you want.

##### Use Case Scenarios

If a ChartAxisLabel is too long and truncated, the tooltip for the label will show the full text. You can also show additional information about the ChartAxisLabel.

##### Adding Tooltip for ChartAxisLabel

To add a tooltip for chart, set the *ShowToolTips* property to true. By default ChartAxisLabel content will be taken as tooltip content. You can also customize the tooltip content using the *ChartFormatAxisLabel* event.

The following code illustrates how to add a customized tooltip for ChartAxisLabel:

[C#]

```
this.chartControl1.ChartFormatAxisLabel += new
ChartFormatAxisLabelEventHandler(chartControl1_ChartFormatAxisLabel);
```

```
ChartSeries series = new ChartSeries("Series");
    series.Points.Add(0,120);
    series.Points.Add(1,220);
    series.Points.Add(2, 150);
    series.Points.Add(3, 240);

    this.chartControl1.Series.Add(series);
//Set labeltext
    arrLabel.Add("India");
    arrLabel.Add("Pakistan");
    arrLabel.Add("Srilanka");
    arrLabel.Add("Japan");
//Set tooltip content
    arrTooltip.Add("IND");
    arrTooltip.Add("PAK");
    arrTooltip.Add("SRL");
    arrTooltip.Add("JPN");
    this.chartControl1.ShowToolTips = true;
    this.chartControl1.Series3D = true;
    this.chartControl1.PrimaryYAxis.Title = "Product sold
(Millions)";
    this.chartControl1.PrimaryXAxis.Title = "Country";
    this.chartControl1.Title.Text = "Product Sales";

void chartControl1_ChartFormatAxisLabel(object sender,
ChartFormatAxisLabelEventArgs e)
{
    int index = (int)e.Value;
    if (e.AxisOrientation == ChartOrientation.Horizontal)
    {
        if (index >= 0 && index < arrLabel.Count)
        {
            e.Label = arrLabel[index].ToString();
//Specify arrTooltip content as chartAxisLabel Tooltip
            e.ToolTip = arrTooltip[index].ToString();
        }
    }
}
```

```
        }  
    }  
    e.Handled = true;  
}
```

**[VB]**

```
AddHandler chartControl1.ChartFormatAxisLabel, AddressOf  
chartControl1_ChartFormatAxisLabel  
  
Dim series As ChartSeries = New ChartSeries("Series")  
    series.Points.Add(0,120)  
    series.Points.Add(1,220)  
    series.Points.Add(2, 150)  
    series.Points.Add(3, 240)  
  
    Me.chartControl1.Series.Add(series)  
  
' Set labeltext  
    arrLabel.Add("India")  
    arrLabel.Add("Pakistan")  
    arrLabel.Add("Srilanka")  
    arrLabel.Add("Japan")  
  
' Set tooltip content  
    arrTooltip.Add("IND")  
    arrTooltip.Add("PAK")  
    arrTooltip.Add("SRL")  
    arrTooltip.Add("JPN")  
    Me.chartControl1.ShowToolTips = True  
    Me.chartControl1.Series3D = True  
  
Me.chartControl1.PrimaryYAxis.Title = "Product sold  
(Millions)"  
Me.chartControl1.PrimaryXAxis.Title = "Country"  
Me.chartControl1.Title.Text = "Product Sales"
```

```
Private Sub chartControll1_ChartFormatAxisLabel(ByVal sender As Object, ByVal e As ChartFormatAxisLabelEventArgs)

    Dim index As Integer = CInt(Fix(e.Value))

    If e.AxisOrientation = ChartOrientation.Horizontal Then

        If index >= 0 AndAlso index < arrLabel.Count Then

            e.Label = arrLabel(index).ToString()

        ' Specify arrTooltip content as chartAxisLabel ToolTip

            e.ToolTip = arrTooltip(index).ToString()

        End If

    End If

    e.Handled = True

End Sub
```

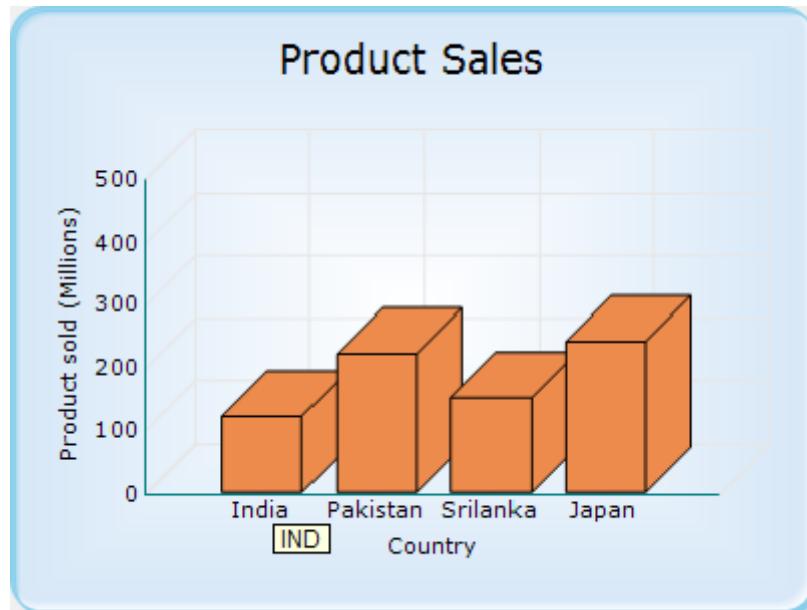


Figure 263: Customized Tooltip

## Refer Also

[Customizing Label Text](#) and [ToolTip](#)

## 4.6.9 Axis Title

Essential Chart provides properties to set custom titles for the axes. Set the title text for an axis using **Title** property. Customize this text using **TitleColor** and **TitleFont** properties.

Chart Axis Property	Description
TitleColor	Sets the color for the title text of the axis.
TitleFont	Sets the font style for the title text.

### [C#]

```
//Sets custom title for x- axis.
this.chartControl1.PrimaryXaxis.Title = "x-axis";
this.chartControl1.PrimaryXaxis.TitleColor = Color.Red;
this.chartControl1.PrimaryXaxis.TitleFont = new Font("Arial", 10);
//Set custom title for y-axis in the similar method.
```

### [VB]

```
'Sets custom title for x- axis.
Me.chartControl1.PrimaryXaxis.Title = "x-axis"
Me.chartControl1.PrimaryXaxis.TitleColor = Color.Red
Me.chartControl1.PrimaryXaxis.TitleFont = New Font("Arial", 10)
'Set custom title for y-axis in the similar method.
```

### Multiline Chart Axes Title

You can now wrap the axes titles and display them as multiline text. Set multiline title text in **Axis.Title** property through designer as follows. Press ENTER key to begin a new line. Press CTRL+ENTER to set the text entered.

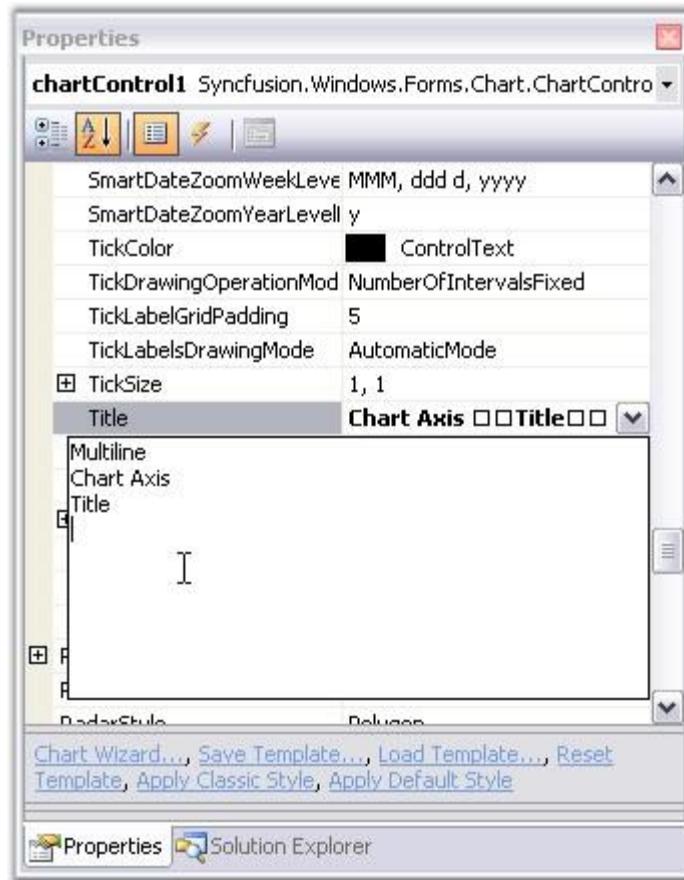


Figure 264: Setting Multiline Axis Title Through Properties Window

The below screenshot illustrates a chart with multiline axes titles.

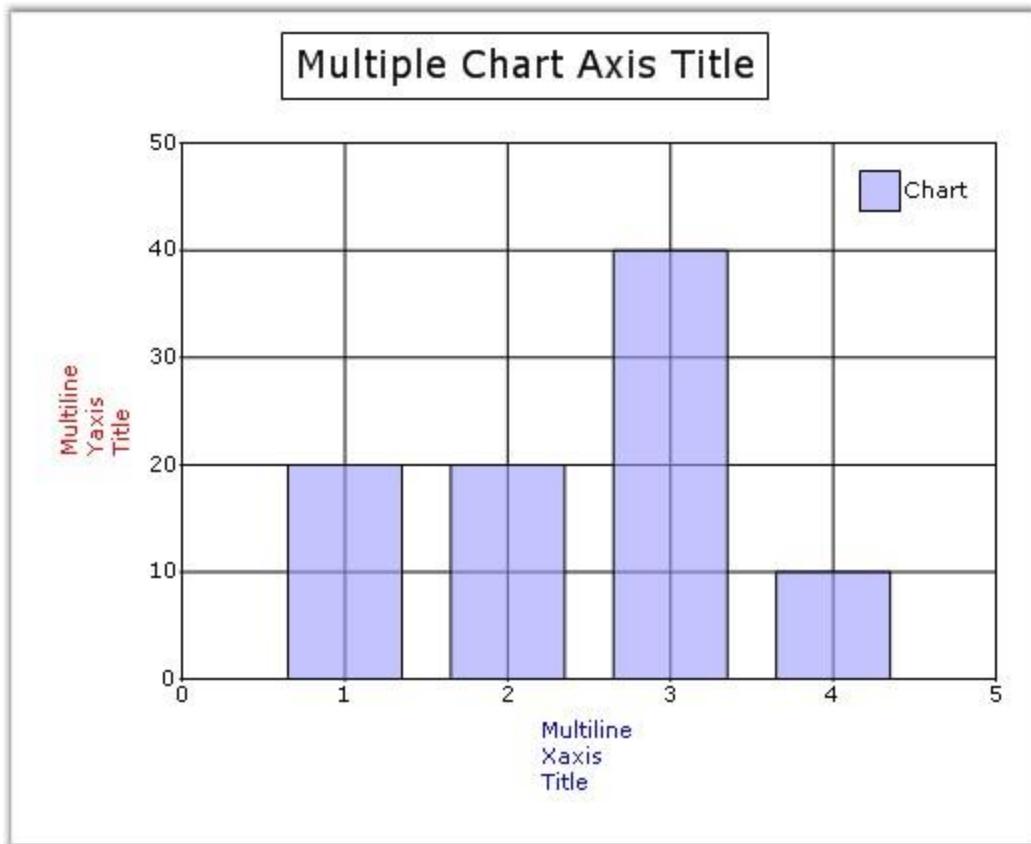


Figure 265: Chart with Multiline Axes Titles

### Drawing Mode of Title Text

You can now display partial axis title with an ellipsis at the end of text, whose text length exceeds the axis length. There is also an option to wrap the title text, in addition to the multiline axes title feature, which is discussed above. The **Axes.TitleDrawMode** property is used to control this behavior.

Chart Axis Property	Description
TitleDrawMode	Sets the drawing mode of the axis title. It can be Ellipse, Wrap or None. By default it is set to <b>None</b> .

[C#]

```
//Setting drawing mode of y-axis title
this.chartControl1.PrimaryXAxis.TitleDrawMode =
ChartTitleDrawMode.Ellipsis;
//Setting drawing mode of secondary y-axis title
```

```
this.secYAxis.TitleDrawMode = ChartTitleDrawMode.Wrap;
```

[VB]

```
'Setting drawing mode of y-axis title
Me.chartControll.PrimaryXAxis.TitleDrawMode =
ChartTitleDrawMode.Ellipsis
'Setting drawing mode of secondary y-axis title
Me.secYAxis.TitleDrawMode = ChartTitleDrawMode.Wrap
```

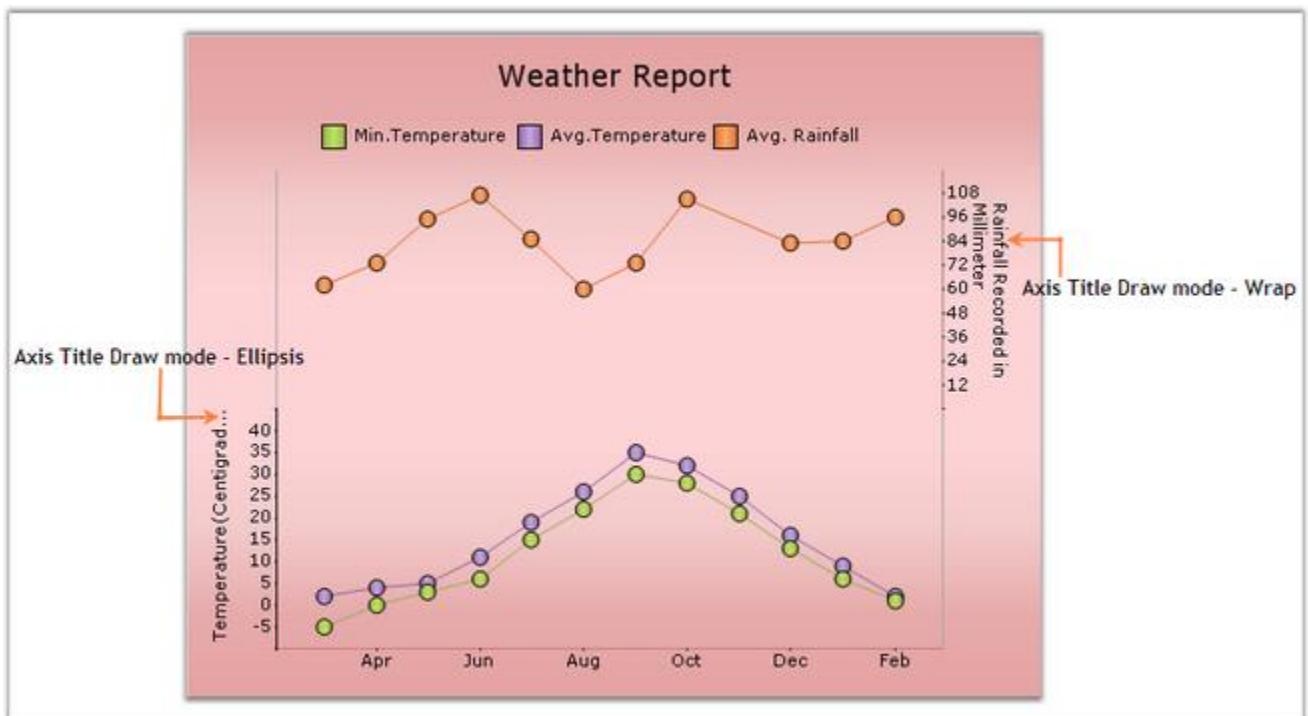


Figure 266: Y-Axis TitleDrawMode = "Ellipsis"; SecYAxis TitleDrawMode = "Wrap"

## 4.6.10 Axis Ticks

### Major Ticks

Major Ticks are rendered automatically at the intersection of an axis with the interval grid lines. Here are some properties that will let you customize the look and feel, and behavior of the ticks.

ChartAxis Property	Description
--------------------	-------------

TickSize	Specifies the width and height of the tick rectangle. This is also a good way to hide the ticks. Default is {1, 1}.
TickCount	Color of the tick mark. Default is <b>System.ControlText</b> .
TickLabelGridPadding	The padding between the tickmark in the axis and the label. Default is 5.
TickDrawingOperationMode	Defines the number of ticks to render while zooming. <ul style="list-style-type: none"> <li>• <b>NumberOfIntervalsFixed</b> - When you zoom, the number of visible intervals will be constant. So, as you zoom in, the total number of intervals will increase.</li> <li>• <b>IntervalFixed</b> - The number of intervals will be constant. So, as you zoom in, fewer intervals will be visible at a time.</li> </ul>

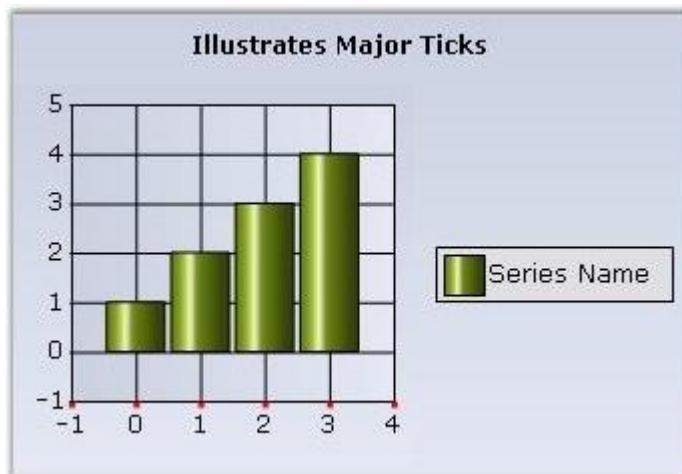


Figure 267: Primary X-Axis with Major ticks (3x3, DarkOrange)

**[C#]**

```
this.chartControl1.PrimaryXAxis.TickSize = new Size(3,3);
this.chartControl1.PrimaryXAxis.TickColor = Color.DarkOrange;
this.chartControl1.PrimaryXAxis.TickLabelGridPadding = 8F;
```

**[VB]**

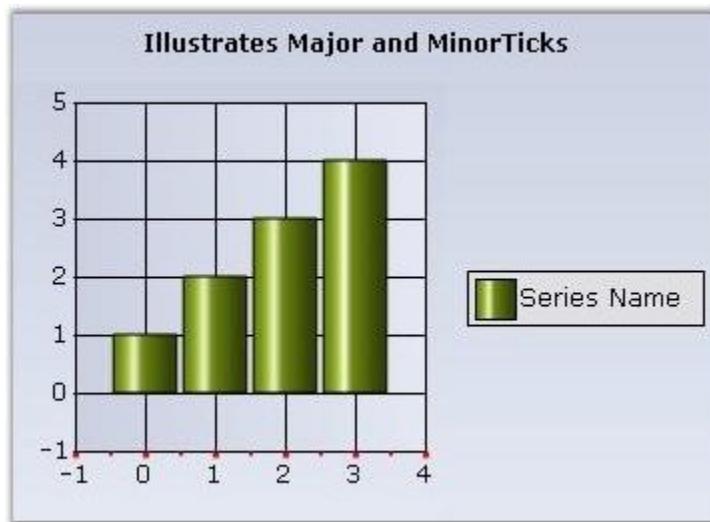
```
Me.chartControl1.PrimaryXAxis.TickSize = New Size(3,3)
```

```
Me.chartControll1.PrimaryXAxis.TickColor = Color.DarkOrange  
Me.chartControll1.PrimaryXAxis.TickLabelGridPadding = 8F
```

### Minor Ticks

Minor ticks are tick marks in between major ticks. These are not rendered by default. Use the properties below to enable and define the frequency of such minor tick marks.

ChartAxis Property	Description
SmallTicksPerInterval	Specifies if and how many minor ticks, which are tick marks drawn on the axis between intervals, should be drawn.
SmallTickSize	Specifies the size of the tick rectangle.



*Figure 268: Primary X-Axis with Major and Minor Ticks*

#### [C#]

```
this.chartControll1.PrimaryXAxis.SmallTickSize = new  
System.Drawing.Size(2, 2);  
this.chartControll1.PrimaryXAxis.SmallTicksPerInterval = 1;
```

#### [VB]

```
this.chartControll1.PrimaryXAxis.SmallTickSize = new  
System.Drawing.Size(2, 2)  
this.chartControll1.PrimaryXAxis.SmallTicksPerInterval = 1
```

#### 4.6.11 3-D Related

Here are some properties that affect the rendering of an axis when in **3-D** mode, which is set using the **Series3D** property.

Chart Control Property	Description
Series3D	Specifies if the chart should be rendered in 3-D mode.
RealMode3D	Specifies if the chart should be rendered in a 3-D plane.
Depth	Specifies the depth of the axes in the z coordinate. Default value is <b>50f</b> .
Tilt	Specifies the tilt angle relative to the y-axis. Default value is <b>30f</b> .
Rotation	Specifies the angle of rotation relative to the x-axis. Default value is <b>30f</b> .
ColumnDrawMode	Specifies the mode of column drawing when in 3-D. <ul style="list-style-type: none"> <li>• <b>PlaneMode</b>—Columns from different series are drawn with the same depth.</li> <li>• <b>InDepthMode</b>—Columns from different series are drawn with different depths.</li> </ul>
EnableMouseRotation	Enables rotation of the chart at runtime using middle/right mouse button.

#### 3D Mode Sample

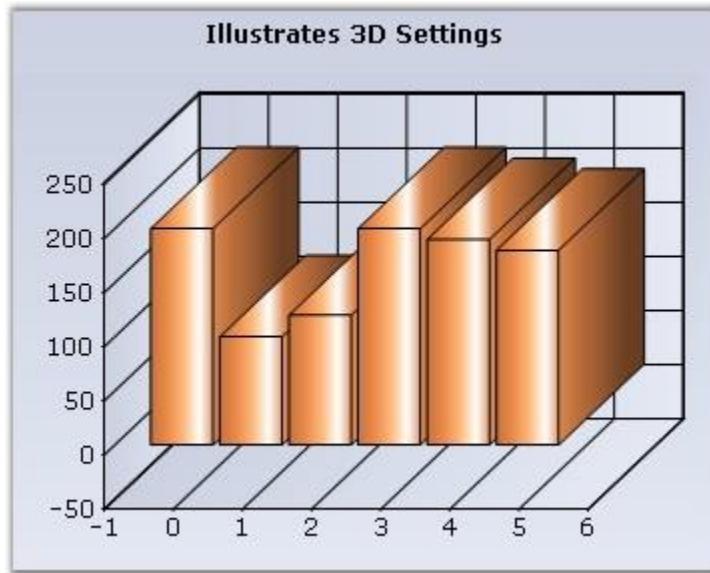
##### [C#]

```
this.chartControl1.Series3D = true;
this.chartControl1.Depth = 55F;
this.chartControl1.Tilt = 55F;
this.chartControl1.Rotation = 60;
```

##### [VB .NET]

```
Me.chartControl1.Series3D = True
Me.chartControl1.Depth = 55F
```

```
Me.chartControl1.Tilt = 55F  
Me.chartControl1.Rotation = 60
```



*Figure 269: 3-D Chart control with Depth = "55f", Tilt = "55f"; Rotation = "60"*

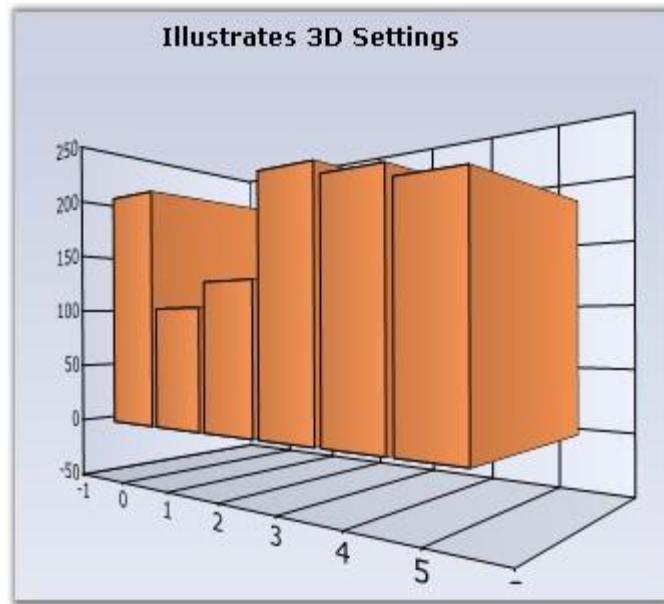
### **Real 3D Mode sample**

**[C#]**

```
this.chartControl1.ChartArea.Series3D = true;  
this.chartControl1.Tilt = 0;  
this.chartControl1.Depth = 150;  
this.chartControl1.Rotation = 10;  
this.chartControl1.RealMode3D = true;
```

**[VB .NET]**

```
Me.chartControl1.ChartArea.Series3D = True  
Me.chartControl1.Tilt = 0  
Me.chartControl1.Depth = 150  
Me.chartControl1.Rotation = 10  
Me.chartControl1.RealMode3D = True
```



*Figure 270: 3D Chart in a 3-D plane with Tilt = "0"; Depth = "150"; Rotation = "10"*

### **Rotating Chart**

The end-users can be allowed to rotate the chart at run-time, using the mouse (middle or right mouse button) by setting the **EnableMouseRotation** property to **true**.



**Note:** *Rotation will not be possible with the LEFT-MOUSE button by enabling this property.*

#### **[C#]**

```
this.chartControl1.RealMode3D = true;  
this.chartControl1.EnableMouseRotation = true;
```

#### **[VB .NET]**

```
Me.chartControl1.RealMode3D = True  
Me.chartControl1.EnableMouseRotation = True
```

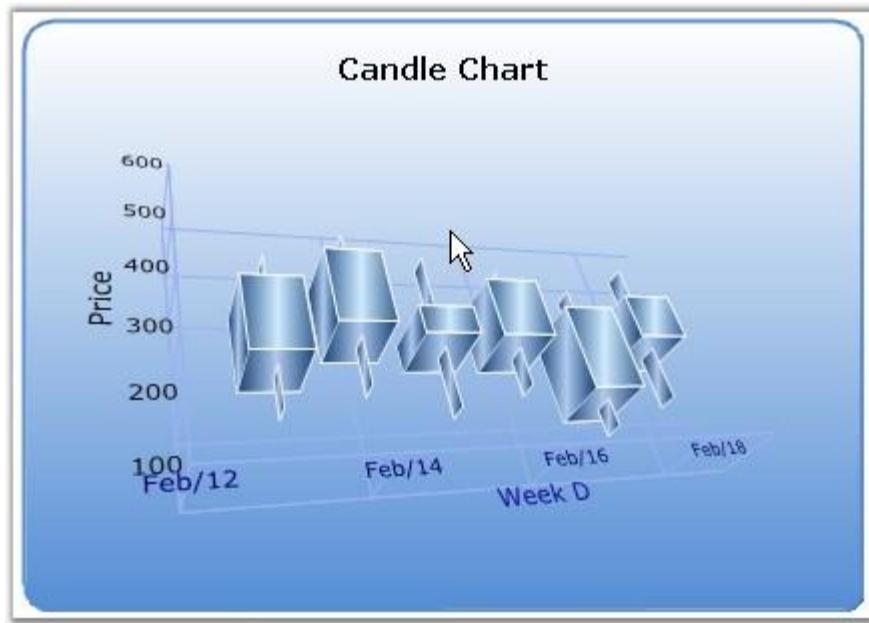


Figure 271: Real3D Mode Chart being Rotated By Using Mouse

### Rendering Chart in 3-D Style

The interior walls of 3-D charts are enhanced with color effects to improve the chart appearance for many 3-D chart types such as column, column range, bar, area, spline area, step area, candle, and histogram. The chart can be rendered in 3-D style using the **Style3D** property. The following table explains this property.

Property	Description
Style3D	Specifies whether the chart should be rendered in 3-D style.

The following code example shows how to enable rendering the chart in 3-D style.

[C#]

```
this.chartControl1.Style3D = true;
```

[VB .NET]

```
Me.chartControl1.Style3D=True
```

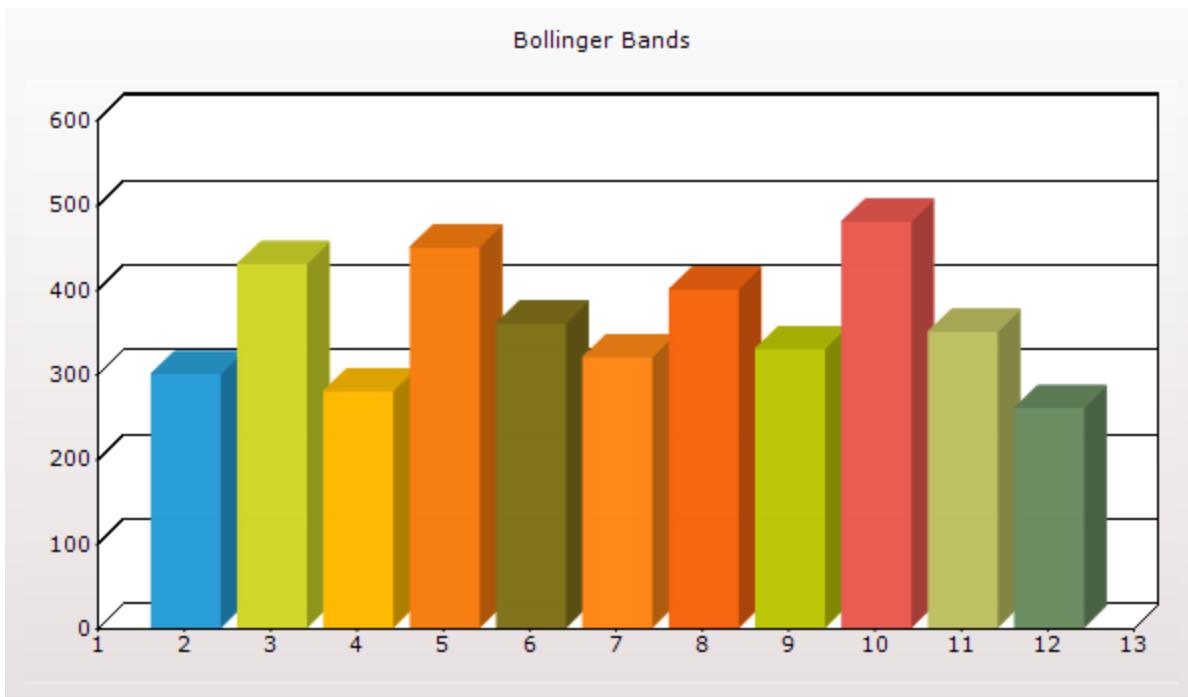


Figure 272: Column Chart with 3-D Style

#### 4.6.12 Chart Grid Lines

The grid lines in the chart that delineates the intervals in the axes can be customized using the following properties.

Chart Control Property	Description
DrawGrid	Specifies whether or not to draw the grid lines.
GridLineType.ForeColor	The forecolor of the line.
GridLineType.BackColor	The back color of the line.
GridLineType.DashStyle	The <b>DashStyle</b> to use for drawing the line.
GridLineType.PenType	The <b>PenType</b> to use for drawing the line.
GridLineType.Width	The thickness of the lines.

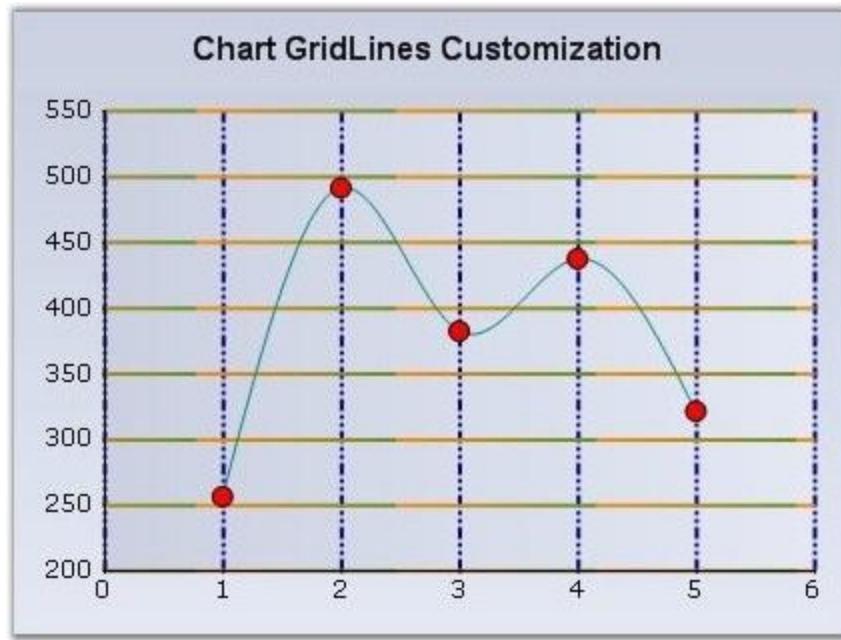


Figure 273: Axes Gridlines customized with Properties

Using the **GridLineType** property, **BackColor**, **DashStyle**, **ForeColor**, **PenType** and the **Width** can be specified.

The following code snippet illustrates how to show the gridlines on both axes and how to customize them.

[C#]

```
//Customizing X-Axis Gridlines
this.chartControl1.PrimaryXAxis.DrawGrid = true;
this.chartControl1.PrimaryXAxis.GridLineType.BackColor =
System.Drawing.Color.Transparent;
this.chartControl1.PrimaryXAxis.GridLineType.DashStyle =
System.Drawing.Drawing2D.DashStyle.DashDotDot;
this.chartControl1.PrimaryXAxis.GridLineType.ForeColor =
System.Drawing.Color.DarkBlue;
this.chartControl1.PrimaryXAxis.GridLineType.Width = 2F;

//Customizing Y-Axis Gridlines
this.chartControl1.PrimaryYAxis.DrawGrid = true;
this.chartControl1.PrimaryYAxis.GridLineType.BackColor =
System.Drawing.Color.OliveDrab;
this.chartControl1.PrimaryYAxis.GridLineType.ForeColor =
System.Drawing.Color.DarkOrange;
```

```
this.chartControl1.PrimaryYAxis.GridLineType.PenType =
System.Drawing.Drawing2D.PenType.LinearGradient;
this.chartControl1.PrimaryYAxis.GridLineType.Width = 2F;
```

## [VB.NET]

```
'Customizing X-Axis Gridlines
Me.chartControl1.PrimaryXAxis.DrawGrid = True
Me.chartControl1.PrimaryXAxis.GridLineType.BackColor =
System.Drawing.Color.Transparent
Me.chartControl1.PrimaryXAxis.GridLineType.DashStyle =
System.Drawing.Drawing2D.DashStyle.DashDotDot
Me.chartControl1.PrimaryXAxis.GridLineType.ForeColor =
System.Drawing.Color.DarkBlue
Me.chartControl1.PrimaryXAxis.GridLineType.Width = 2F

'Customizing Y-Axis Gridlines
Me.chartControl1.PrimaryYAxis.DrawGrid = True
Me.chartControl1.PrimaryYAxis.GridLineType.BackColor =
System.Drawing.Color.OliveDrab
Me.chartControl1.PrimaryYAxis.GridLineType.ForeColor =
System.Drawing.Color.DarkOrange
Me.chartControl1.PrimaryYAxis.GridLineType.PenType =
System.Drawing.Drawing2D.PenType.LinearGradient
Me.chartControl1.PrimaryYAxis.GridLineType.Width = 2F
```

### 4.6.13 Chart StripLines

Strip-lines are bands that are drawn at the background of the chart. They can be used to highlight areas of interest. They can be either vertical or horizontal and may be specified with a variety of options to precisely control where they are placed and how they are repeated. The strip-lines are stored in the **ChartAxis.StripLines** collection, which holds objects of class **ChartStripLine**.

A strip-line is configurable by setting its start, end, period and width in the same value type as the axis that holds it. The interior of the strip-lines support gradients, images and different text positions and orientations.

Chart control Property	Description
BackImage	Sets the background image for the stripline.
DateOffset	Gets / sets the offset of the stripline if the chart's PrimaryX-axis is of type <b>Datetime</b> and StartAtAxisPosition is <b>true</b> . Also see Offset.

Enabled	Enables the Stripline.
End	Gets /sets the end range (in double) of the stripline. Use this if the axis range type is <b>Double</b> . Also see EndDate.
EndDate	The end date of the stripline. Use this if the axis range type is <b>DateTime</b> . Also see End.
Font	The Font style in the which the stripline text if any will be rendered.
FixedWidth	Specifies a fixed width for the chart stripline. Normally, the width of the stripline changes when the axis range changes. You can also set the width to be fixed irrespective of the AxisRange, by specifying a width in this property. After setting a fixed width, the stripline width will not vary beyond / less than the value that is set.
Interior	Interior brush information for the stripline.
Offset	Gets / sets the offset of the stripline if the chart's PrimaryX-axis is of type <b>Double</b> and StartAtAxisPosition is <b>true</b> . Also see DateOffset.
Period	Gets / sets the period (width of the range) over which the stripline appears.
PeriodDate	Gets / sets the period (time span) over which the stripline appears if the value is <b>DateTime</b> .
Start	Gets / sets the start of the stripline. Also see End.
StartAtAxisPosition	Indicates whether the Stripline will start at the start of the axis range.
StartDate	The start date of the stripline.
Text	The text in the stripline.
TextAlignment	Alignment of the text in the stripline.
TextColor	The color of the text in the stripline.
Vertical	Indicates whether stripline is rendered vertically.
Width	The width of the stripline.
WidthDate	Gets / sets the width of the stripline in a Time span.

The following is the code to draw a stripline from x-axis with **DateTime** values.

**[C#]**

```
//Declaring
ChartStripLine stripLine = new ChartStripLine();

//Customizing the Stripline
stripLine.Enabled = true;
stripLine.Vertical = false;
stripLine.Start = 140;
stripLine.Width = 35;
stripLine.FixedWidth = 30;
stripLine.End = 175;
stripLine.Text = "100% of Quota";
stripLine.TextColor = Color.Cyan;
stripLine.TextAlignment = ContentAlignment.MiddleCenter;
stripLine.Font = new Font("Arial", 10, FontStyle.Bold);
stripLine.Interior = new BrushInfo(230, new
BrushInfo(GradientStyle.Vertical, Color.OrangeRed, Color.DarkKhaki));

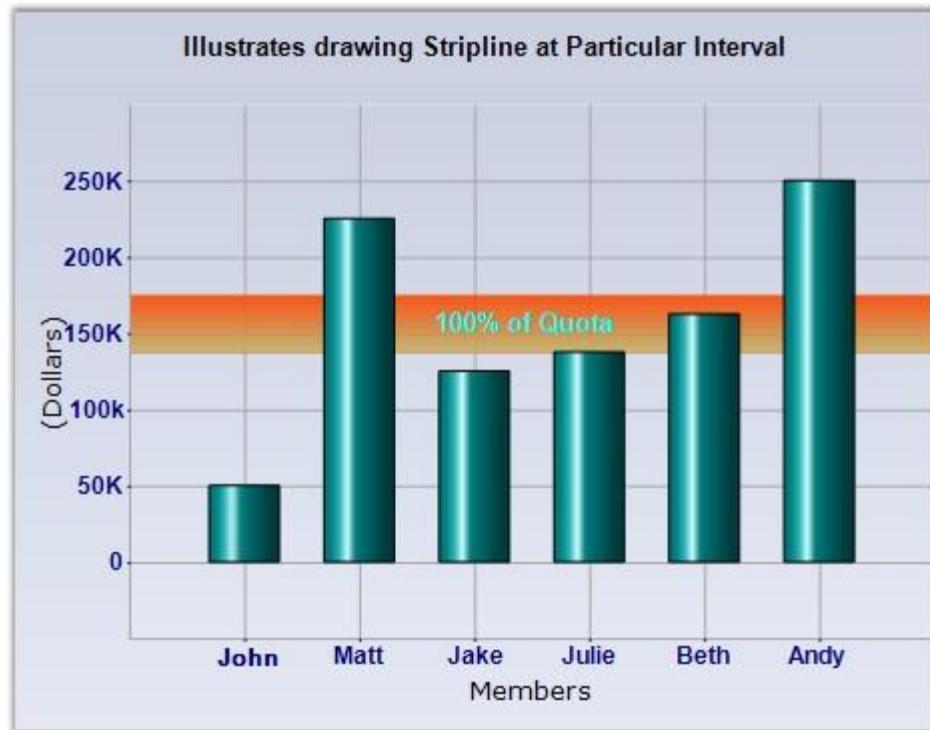
//Adding stripline to the X-axis
this.chartControl1.PrimaryYAxis.StripLines.Add(stripLine);
```

**[VB .NET]**

```
'Declaring
Private stripLine As ChartStripLine = New ChartStripLine()

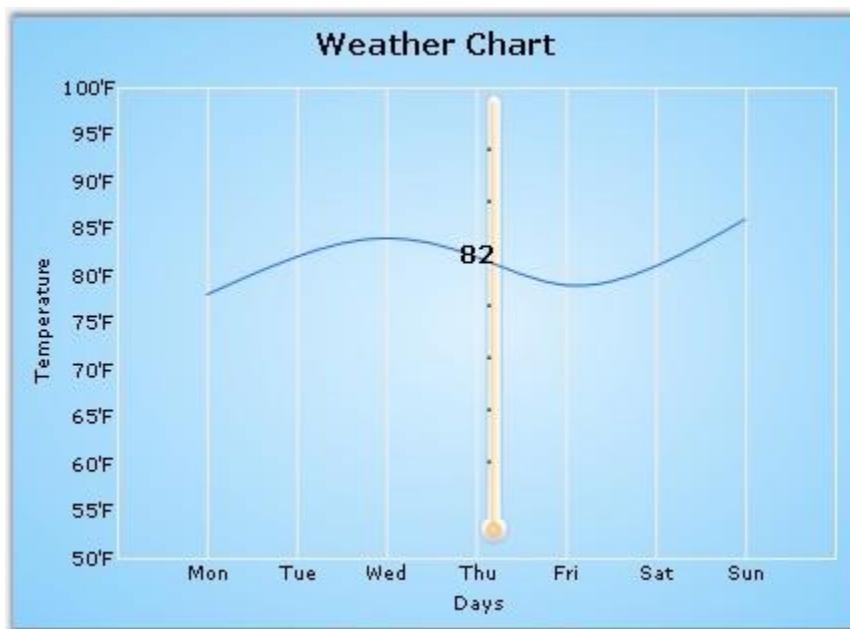
'Customizing the Stripline
stripLine.Enabled = True
stripLine.Vertical = True
stripLine.Start = 140
stripLine.Width = 35
stripLine.FixedWidth = 30
stripLine.End = 175
stripLine.Text = "100% of Quota"
stripLine.TextColor = Color.Cyan
stripLine.TextAlignment = ContentAlignment.MiddleCenter
stripLine.Font = New Font("Arial",10,FontStyle.Bold)
stripLine.Interior = New BrushInfo(230, new
BrushInfo(GradientStyle.Vertical,Color.OrangeRed, Color.DarkKhaki))

'Adding stripline to the X-axis
Me.chartControl1.PrimaryXAxis.StripLines.Add(stripLine)
```



*Figure 274: Chart with StripLine rendered in Y-Axis*

Use an image as StripLine by setting through **StripLine.BackImage** property.



*Figure 275: Chart with Image StripLine rendered in X-Axis*

## 4.6.14 Chart Breaks

Breaks are very useful if you add points with too large difference in values. To enable breaks, you need to set the **ChartAxis.MakeBreaks** property to **true** and set the break mode (ChartAxis.BreakRanges.BreaksMode property).

There are three possible modes. They are,

- **ChartBreaksMode.None** - If this value is set, breaks are not used.
- **ChartBreaksMode.Manual** (default) - If this value is set, you can manually set the breaks ranges. To do this, use following methods.
  - **ChartAxis.BreakRanges.Union** – add a new break range.
  - **ChartAxis.BreakRanges.Exclude** – remove the break range.
  - **ChartAxis.BreakRanges.Clear** – remove all break ranges.
- **ChartBreaksMode.Auto** - If this mode is enabled, chart will compute the breaks ranges automatically. You can use the ChartAxis.BreakRanges.BreakAmount to set the minimal relative difference between values (default value is 0.1, value range is 0.1). The ratio of empty space should be less than the property value to break the range.
- This mode has several exclusions.
- Breaks are computed only for actual y-axis of series.
- Breaks don't work with zooming.
- Breaks don't work with stacking.

All breaks work only with decart axes.

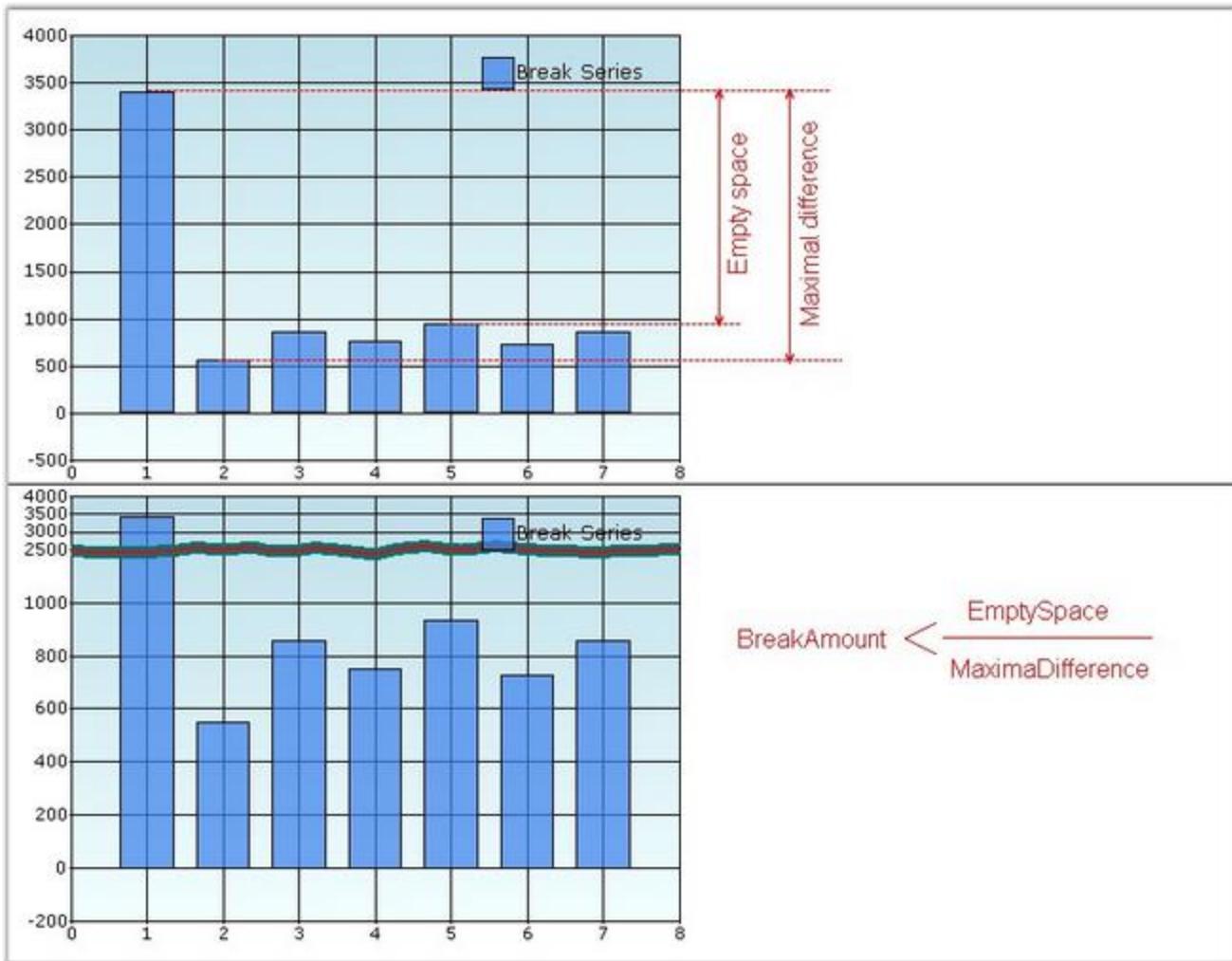


Figure 276: Illustrates Chart BreakAmount Value

[C#]

```
this.chartControl1.PrimaryYAxis.MakeBreaks = true;

this.chartControl1.PrimaryYAxis.BreakRanges.BreaksMode =
ChartBreaksMode.Manual;
this.chartControl1.PrimaryYAxis.BreakRanges.Union(new DoubleRange(500,
600));
this.chartControl1.PrimaryYAxis.BreakRanges.Union(new DoubleRange(950,
3000));

this.chartControl1.PrimaryYAxis.BreakInfo.LineType =
ChartBreakLineType.Wave;
this.chartControl1.PrimaryYAxis.BreakInfo.LineSpacing = 5;
```

```
this.chartControl1.PrimaryYAxis.BreakInfo.LineColor = Color.Black;
this.chartControl1.PrimaryYAxis.BreakInfo.LineWidth = 1;
this.chartControl1.PrimaryYAxis.BreakInfo.LineStyle = DashStyle.Dot;
this.chartControl1.PrimaryYAxis.BreakInfo.SpacingColor = Color.White;
this.chartControl1.PrimaryYAxis.BreakRanges.BreakAmount = 0.5;
```

**[VB .NET]**

```
Me.chartControl1.PrimaryYAxis.MakeBreaks = True

Me.chartControl1.PrimaryYAxis.BreakRanges.BreaksMode =
ChartBreaksMode.Manual
Me.chartControl1.PrimaryYAxis.BreakRanges.Union(New DoubleRange(500,
600))
Me.chartControl1.PrimaryYAxis.BreakRanges.Union(New DoubleRange(950,
3000))

Me.chartControl1.PrimaryYAxis.BreakInfo.LineType =
ChartBreakLineType.Wave
Me.chartControl1.PrimaryYAxis.BreakInfo.LineSpacing = 5
Me.chartControl1.PrimaryYAxis.BreakInfo.LineColor = Color.Black
Me.chartControl1.PrimaryYAxis.BreakInfo.LineWidth = 1
Me.chartControl1.PrimaryYAxis.BreakInfo.LineStyle = DashStyle.Dot
Me.chartControl1.PrimaryYAxis.BreakInfo.SpacingColor = Color.White
Me.chartControl1.PrimaryYAxis.BreakAmount = 0.5
```

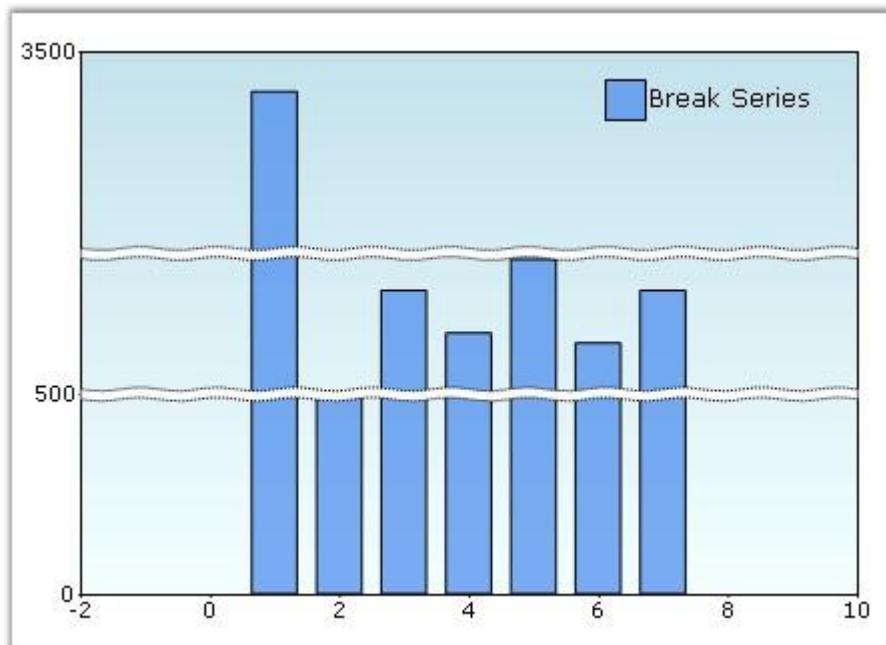


Figure 277: Chart BreakAmount = "0.5"

## 4.6.15 Axis Crossing Support

Essential Chart for Windows allows the X and Y axis to intersect at a desired point. The X and Y axis will intersect at a point based on the value specified in the *X axis Crossing* property and the *Y axis Crossing* property respectively.

### Use Case Scenarios

This feature will be useful to customize the location of primary axes from default location, when you want to add huge number of negative and positive points in the chart.

### Properties

Table 2: Property Table

Property	Description	Type	Data Type	Reference links
Crossing	Specifies the point of intersect for X and Y axis based on the given data point value.	Server side	Double	NA

### Sample Link

To view a sample:

1. Open the **Syncfusion Dashboard**.
2. Click the Windows Forms drop-down list and select **Run Locally Installed Samples**.
3. Navigate to **Chart samples → Chart Axes → Axis Crossing**.

### Enable crossing X and Y axis

To enable crossing X and Y axis, specify the Y axis data point value, where you want the X axis to cross, in the *X axis Crossing* property. Similarly specify the X axis data point value, where you want the Y axis to cross, in the *Y axis Crossing* property. The following code illustrates this:

[C#]

```
this.chartControl1.PrimaryXAxis.Crossing=150;
```

```
this.chartControl1.PrimaryYAxis.Crossing = 6;  
  
this.chartControl1.Series3D = true;
```

[VB]

```
Me.chartControl1.PrimaryXAxis.Crossing = 150  
Me.chartControl1.PrimaryYAxis.Crossing = 6  
Me.chartControl1.Series3D = True
```

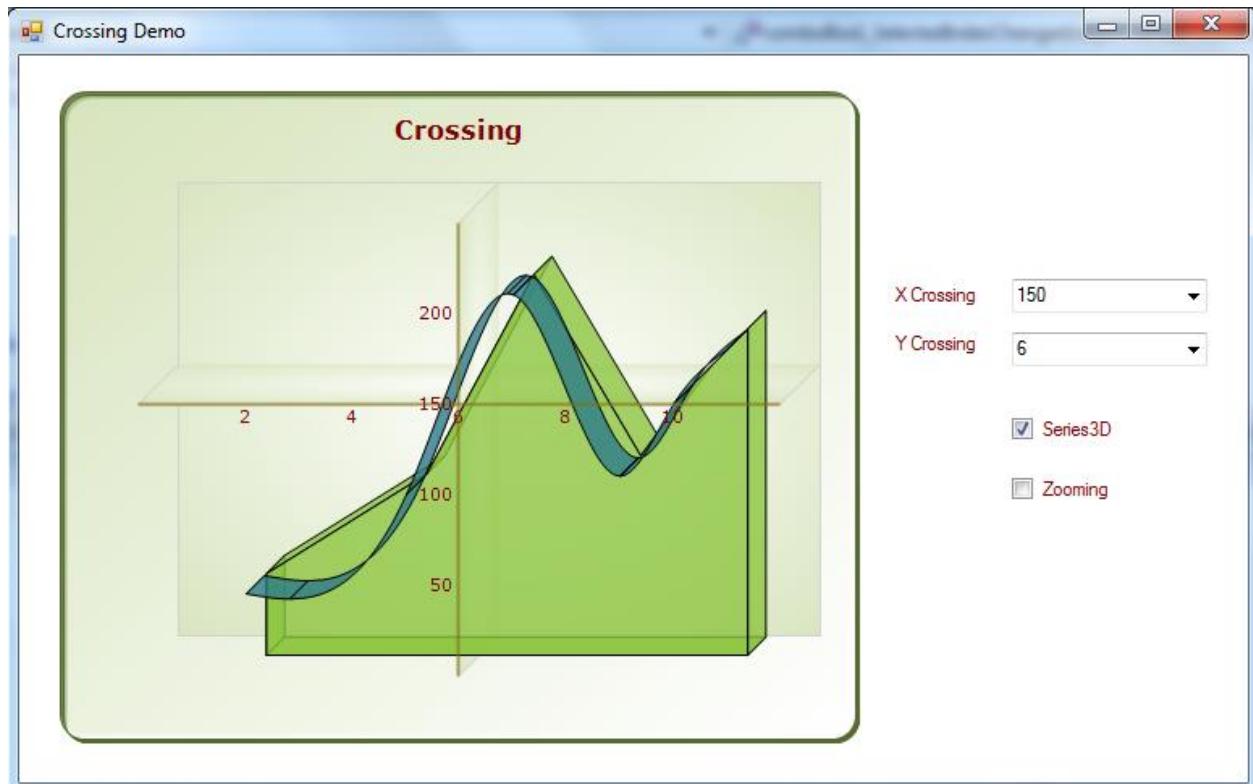


Figure 278: Primary axes location customized

#### 4.6.16 Axis Label Placement

This feature enables you to specify the position of the label for an axis. You can place the label either inside or outside the plotted chart area.

## Use Case Scenarios

When you have lengthy label for **the** chart axis, it will occupy more space. So the plotted chart area will **get** reduced. You can avoid this using this feature.

## Properties

Table 3: Property Table

Property	Description	Type	Data Type
AxisLabelPlacement	Specifies the position of the label in a chart axes.  It can be placed inside or outside the plotted chart area using <b>ChartPlacement</b> enum.	NA	NA

## Sample Link

To view a sample:

1. Open the **Syncfusion Dashboard**.
2. Click the **User Interface > Windows Forms**.
3. Click **Run Samples**.
4. Navigate to **Chart samples > Chart Axes > ChartAxisCustomization**.

## Positioning Axis Label

You can position the chart axis label using the **Axes.AxisLabelPlacement** property. You can specify whether the axis label should be placed inside or outside the plotted chart area using the **ChartPlacement** enum.

The following code illustrates how to place the chart axis label inside the plotted chart area:

[C#]

```
this.chartControl1.PrimaryXAxis.AxisLabelPlacement = ChartPlacement.Inside;
```

[VB]

```
Me.chartControl1.PrimaryXAxis.AxisLabelPlacement = ChartPlacement.Inside
```

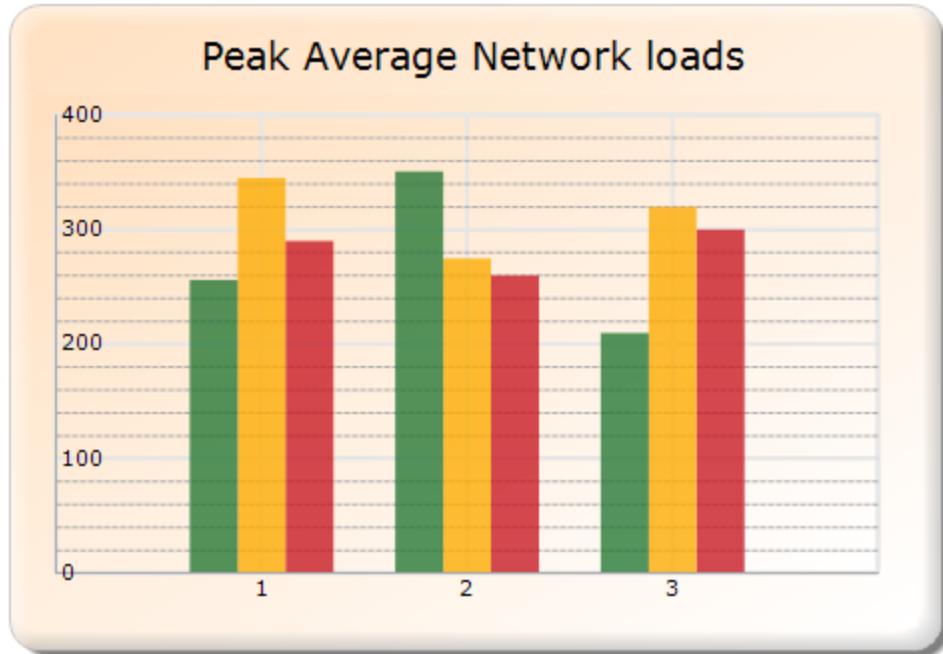


Figure 279: Chart with Y-Axis Labels Placedmed Inside of Axis

### Positioning Individual Axis Labels

Essential Chart supports customizing the individual axis label position to the right or left side of the axis for a horizontal axis, and to the top or bottom for a vertical axis based on the user's needs.

This feature can be achieved by using the **ChartFormatAxisLabel** event of the chart axis. The following code illustrates customizing the individual label position by using **AxisLabelPlacement** in the **ChartFormatAxisLabel** event.

```
[C#]
this.chartControl1.PrimaryYAxis.AxisLabelPlacement =
ChartPlacement.Inside;

//Sets the AxisLabelPlacement property in the ChartFormatAxisLabel event.

private void chartControl1_ChartFormatAxisLabel(object sender,
ChartFormatAxisEventArgs e)
{
    if (e.AxisOrientation == ChartOrientation.Vertical)
    {
        if (e.Label == "1")
        {
            if (series.Points[(int)e.Value - 1].YValues[0] > 0)
                e.AxisLabelPlacement =
ChartPlacement.Outside;
        }
    }
}
```

```

        e.Label = "Canada";
    }
    else if (e.Label == "2")
    {
        if (series.Points[(int)e.Value - 1].YValues[0] > 0)
            e.AxisLabelPlacement =
ChartPlacement.Outside;
        e.Label = "France";
    }
    else if (e.Label == "3")
    {
        if (series.Points[(int)e.Value - 1].YValues[0] > 0)
            e.AxisLabelPlacement =
ChartPlacement.Outside;
        e.Label = "Japan";
    }
    else if (e.Label == "4")
    {
        if (series.Points[(int)e.Value - 1].YValues[0] > 0)
            e.AxisLabelPlacement = ChartPlacement.Outside;
        e.Label = "Britain";
    }
    else if (e.Label == "5")
    {
        if (series.Points[(int)e.Value - 1].YValues[0] > 0)
            e.AxisLabelPlacement =
ChartPlacement.Outside;
        e.Label = "United States";
    }
    e.Handled = true;
}
}
}

```

**[VB]**

```

Me.chartControl1.PrimaryYAxis.AxisLabelPlacement =
ChartPlacement.Inside

'Sets the AxisLabelPlacement property in ChartFormatAxisLabel event.
this.chartControl1.PrimaryYAxis.AxisLabelPlacement =
ChartPlacement.Inside;

Private Sub chartControl1_ChartFormatAxisLabel(ByVal sender As Object,
ByVal e As ChartFormatAxisLabelEventArgs)
    If e.AxisOrientation = ChartOrientation.Vertical Then
        If e.Label = "1" Then
            If series.Points(CInt(Fix(e.Value)) - 1).YValues(0)
> 0 Then
                e.AxisLabelPlacement = ChartPlacement.Outside
            End If
    End If
End Sub

```

```

        e.Label = "Canada"
    ElseIf e.Label = "2" Then
        If series.Points(CInt(Fix(e.Value)) - 1).YValues(0)
> 0 Then
            e.AxisLabelPlacement = ChartPlacement.Outside
        End If
        e.Label = "France"
    ElseIf e.Label = "3" Then
        If series.Points(CInt(Fix(e.Value)) - 1).YValues(0)
> 0 Then
            e.AxisLabelPlacement = ChartPlacement.Outside
        End If
        e.Label = "Japan"
    ElseIf e.Label = "4" Then
        If series.Points(CInt(Fix(e.Value)) - 1).YValues(0)
> 0 Then
            e.AxisLabelPlacement = ChartPlacement.Outside
        End If
        e.Label = "Britain"
    ElseIf e.Label = "5" Then
        If series.Points(CInt(Fix(e.Value)) - 1).YValues(0)
> 0 Then
            e.AxisLabelPlacement = ChartPlacement.Outside
        End If
        e.Label = "United States"
    End If
    e.Handled = True
End If
End Sub

```

The following screenshot illustrates the customization options for individual label positions on the y-axis to the right or left side based on the y value of the data points. If the export value is positive, the label is rendered to the left side of the axis, and if it is negative, the label is rendered on the right side of the axis.

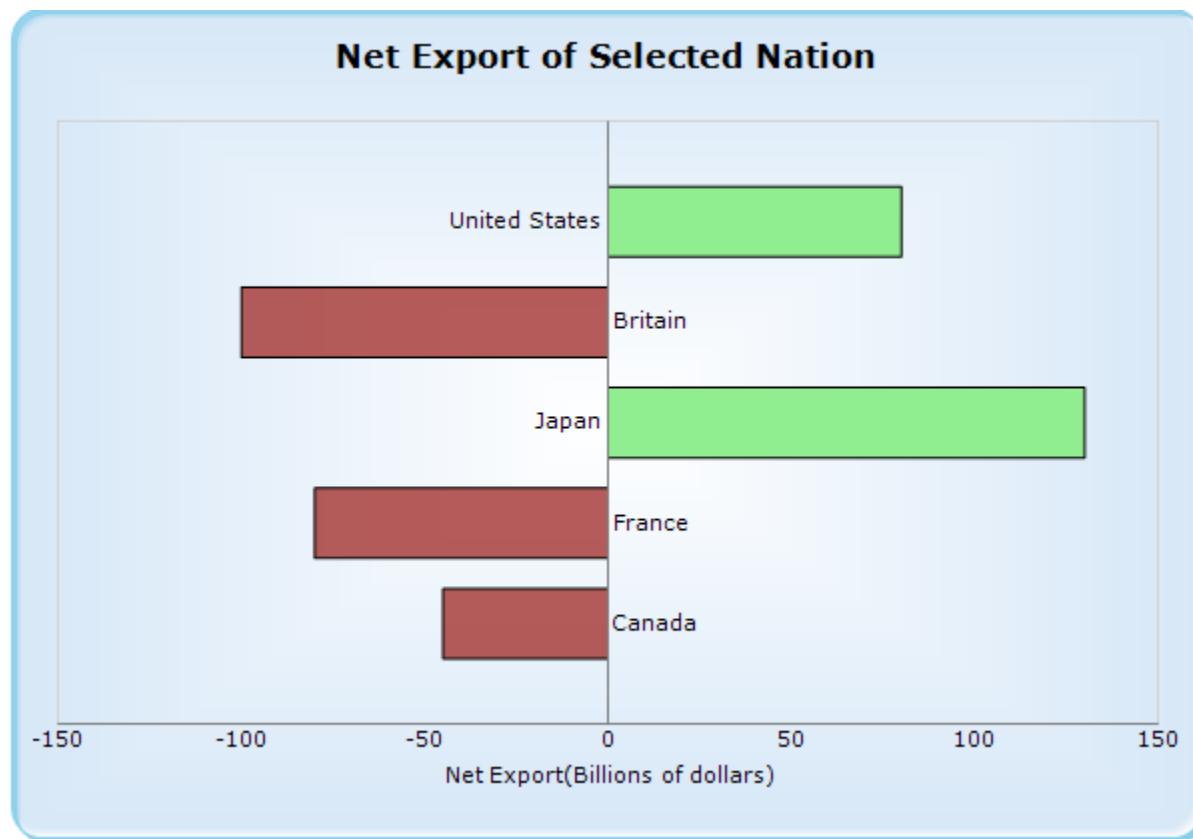


Figure 280: Customized Individual Y-Axis Label Placement

## 4.7 Chart Area

Essential Chart comes with chart divide area support, wherein a single ChartArea can be divided into equal squares to display more than one chart(pie, funnel or pyramid). To enable this **ChartArea.DivideArea** property should be set to **true**.

By enabling this property, the following are possible.

- Retrieving the bounds of each sections of pie, funnel or pyramid charts.
- It is possible to show series name as title for individual section of a pie, funnel and pyramid chart types.
- Draw [pie series with the same radius](#).

**GetSeriesBounds()** method can be used to get the bounds of the DividedArea when ChartArea.DivideArea is set to true.

```
[C#]
```

```
this.chartControl1.ChartArea.GetSeriesBounds(series);
```

**[VB .NET]**

```
Me.chartControl1.ChartArea.GetSeriesBounds(series)
```

**ShowSeriesTitle** property is used to display the series name as title for each section of the pie, funnel, pyramid charts in the divided area.

**[C#]**

```
ChartSeries.ConfigItems.PieItem.ShowSeriesTitle = true;  
ChartSeries.ConfigItems.FunnelItem.ShowSeriesTitle = true;  
ChartSeries.ConfigItems.PyramidItem.ShowSeriesTitle = true;
```

**[VB .NET]**

```
ChartSeries.ConfigItems.PieItem.ShowSeriesTitle = True  
ChartSeries.ConfigItems.FunnelItem.ShowSeriesTitle = True  
ChartSeries.ConfigItems.PyramidItem.ShowSeriesTitle = True
```

A sample which demonstrates the chartdividearea support is available in the following sample installation location.

[<sample installed location>\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Chart Appearance\Chart Divide Area](#)

**See Also**

[PieWithSameRadius](#)

## 4.8 Chart Legend and Legend Items

Essential Chart by default displays a legend with information on each series that has been plotted on the chart.

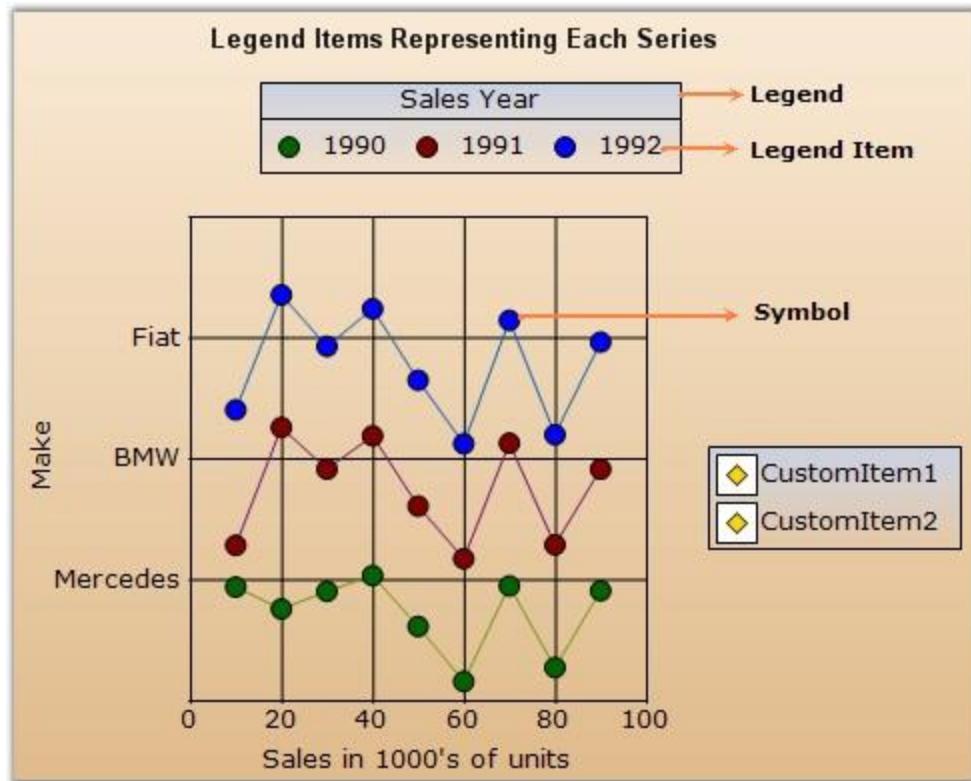


Figure 281: Chart with Legend and Legend Items

1. **Legend** - The rectangular region that lists one or more legend items.
2. **Legend Item** - Represented by an icon or image and a text; this usually gets rendered automatically corresponding to each ChartSeries in the chart. You can also add custom legend items to a Legend.
3. **Symbols** - These refer to the symbols drawn at the data points in a plot. The legend items corresponding to the series can also be rendered with this symbol instead of an icon.

You can turn off the legend by setting the **ShowLegend** property in the chart to **false**. The legend instances in the Chart are exposed via the **Legends** collection. The first entry in this list is considered the "default legend" and is exposed by the **Legend** property.

### 4.8.1 ChartLegend

The legend is represented by the **ChartLegend** type.

#### Default Legend

By default, a custom **ChartLegend** instance gets added to the **Legends** list in the control. You can access this default legend as follows.

**[C#]**

```
// Changing the position of the default legend  
this.chartControl1.Legends[0].LegendPosition =  
Syncfusion.Windows.Forms.Chart.ChartDock.Top;
```

**[VB .NET]**

```
' Changing the position of the default legend  
Me.chartControl1.Legends[0].LegendPosition =  
Syncfusion.Windows.Forms.Chart.ChartDock.Top
```

## Adding Custom Legends

You can add custom legends to the chart through the **Legends** list as follows:

**[C#]**

```
// Changing the position of the default legend  
ChartLegend legend2 = new ChartLegend(chartControl1);  
legend2.Name = "MyLegend";  
chartControl1.Legends.Add(legend2);
```

**[VB .NET]**

```
Dim legend2 As New ChartLegend()  
legend2.Name = "MyLegend"  
chartControl1.Legends.Add(legend2)
```

You can then add custom legend items into the **ChartLegend** through the **CustomItems** property as explained in the next topic ([ChartLegendItem](#)).

You can also associate a **ChartSeries** to a custom **ChartLegend** as follows (then the legend item corresponding to that series will be rendered within the specified legend):

**[C#]**

```
// Associate legend1 with series1
series[0].LegendName = "legend1";
// Associate legend2 with series2
series[1].LegendName = "legend2";
```

**[VB .NET]**

```
' Associate legend1 with series1
series[0].LegendName = "legend1"
' Associate legend2 with series2
series[1].LegendName = "legend2"
```

**Legend Look and Feel**

Here are some common properties you could use to customize the overall legend appearance:

ChartLegend Property	Description
BackColor	Gets / sets the background color of the legend. The default value is <b>Transparent</b> .
VisibleCheckBox	If set to <b>true</b> , a checkbox will be displayed beside each legend item. And if this checkbox is unchecked the corresponding series will disappear from the chart plot. Default is <b>false</b> .
<b>Windows</b>	
Border	Gets / sets the border style of the legend. <b>ShowBorder</b> should be <b>true</b> .
ShowBorder	Specifies whether a border should be drawn. By default it is set to <b>false</b> .
Font	Specifies the font that is to be used for the text rendered in the legend items. The default font style is Verdana, 8, Regular.
BackInterior	Sets the interior appearance for the legend. This overrides the <b>BackColor</b> property.
BackgroundImage	Sets the background image for the legend. This setting overrides the <b>BackInterior</b> property settings.
BackgroundImageLayout	Sets the layout for the background image.

## Legend Positioning

The legend positioning can be affected in the following ways.

ChartLegend Property	Description
Position	<p>Specifies the position relative to the chart at which to render the legend.</p> <ul style="list-style-type: none"> <li>• Top - above the chart</li> <li>• Left - left of the chart</li> <li>• Right - right of the chart</li> <li>• Bottom - below the chart</li> <li>• Floating - will not be docked to any specific location(<b>default setting</b>)</li> </ul>
LegendAlignment	When docked to a side, this property specifies how the legend should be aligned with respect to the chart boundaries.
LegendPlacement	Specifies the placement of a legend in a chart. It can be placed Inside or Outside the chart area using ChartPlacement enum.
DockingFree	If set to <b>true</b> , the legend will be floating and cannot be dragged and docked to the sides.
Behavior	<p>Specifies the docking behavior of the Legend.</p> <ul style="list-style-type: none"> <li>• <b>Docking</b> - It is dockable on all four sides</li> <li>• <b>Movable</b> - It is movable</li> <li>• <b>All</b> - It is movable and dockable</li> <li>• <b>None</b> - It is neither movable nor dockable</li> </ul>
FloatingAutoSize	Specifies whether to determine the size automatically or not, while floating.
OnlyColumnsForFloating	The legend items will be displayed vertically in columns when floating.
RowCount	Specifies the number of rows in which the legend items should be rendered.
ColumnCount	Specifies the number of columns in which the legend items should be rendered.



**Note:** Note that the user can drag the legend around during run time. He can dock it to the sides if docking is enabled. Docking behavior is controlled by Behavior property which is described in the above table.

## Changing Legend Properties at Run Time

The Legend's look and feel can also be customized during runtime. Double-clicking legend's text will pop up the below properties window. Properties set through this dialogue can be applied to the chart.



**Note:** These settings will be lost when the application is closed.

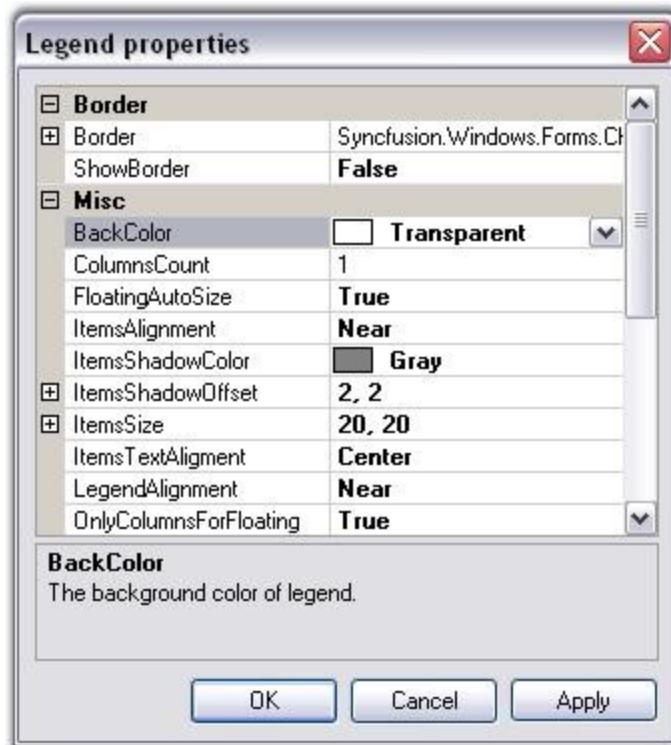


Figure 282: Legend Properties Dialog Box

## See Also

[ChartLegend](#)

### 4.8.2 ChartLegendItem

The legend item is represented by the `ChartLegendItem` type.

#### Default Series LegendItems

Every ChartSeries in the chart control has a ChartLegendItem associated with it. This legend item gets automatically added to the default ChartLegend.

But, if you want to get that associated with a custom ChartLegend, use the LegendName to specify that chart legend as follows:

**[C#]**

```
// Specifies the custom ChartLegend with which this series' legend item  
// should be associated with  
series1.LegendName = "MyLegend";
```

**[VB .NET]**

```
' Specifies the custom ChartLegend with which this series' legend item  
// should be associated with  
series1.LegendName = "MyLegend"
```

### **Adding Custom Legend Items**

To add your own custom legend items to a legend, use the **CustomItems** property in the ChartLegend as follows.

**[C#]**

```
// Adding some custom items into the 2nd custom Legend  
  
ChartLegendItem legendItem1 = new ChartLegendItem();  
legendItem1.ItemStyle.ShowSymbol = true;  
legendItem1.ItemStyle.Symbol.Shape = ChartSymbolShape.Circle;  
legendItem1.ItemStyle.Symbol.Color = Color.Blue;  
legendItem1.Text = "Legend Item";  
  
this.chartControl1.Legends[1].CustomItems = new ChartLegendItem[] {  
    legendItem1};
```

**[VB .NET]**

```
'Adding some custom items into the 2nd custom Legend  
  
Dim legendItem1 As New ChartLegendItem()  
legendItem1.ItemStyle.ShowSymbol = True  
legendItem1.ItemStyle.Symbol.Shape = ChartSymbolShape.Circle  
legendItem1.ItemStyle.Symbol.Color = Color.Blue
```

```
legendItem1.Text = "Legend Item"

'Adding the custom Legend item to the chart
Me.chartControl1.Legends[1].CustomItems = New ChartLegendItem()
{legendItem1}
```



Figure 283: Custom Legend Item

### Customizing items through event

There is also a way to specify custom legend item via events right before they get rendered.

In this example, we reverse the order in which the legend items are rendered through the **FilterItems** event.

[C#]

```
private void Legend_FilterItems(object sender,
ChartLegendFilterEventArgs e)
{
    //This creates an new instance of the ChartLegendItemCollection
    ChartLegendItemsCollection items = new
    ChartLegendItemsCollection();
    for (int i = e.Items.Count - 1; i >= 0; i--)
        items.Add(e.Items[i]);
    e.Items = items;
}
```

[VB.NET]

```
Private Sub Legend_FilterItems(ByVal sender As Object, ByVal e As
ChartLegendFilterEventArgs)
    'This creates an new instance of the ChartLegendItemCollection
    Dim item As New ChartLegendItemsCollection()
    For i As Integer = e.Items.Count - 1 To 0 Step -1
        item.Add(e.Items(i))
    Next
    e.Items = item
End Sub
```



Figure 284: Chart with Legends in specified order using FilterItems Event

### Legend Item's Look and Feel

The legend item's look and feel can be customized to a good extent using the following properties in ChartLegend.

These settings affect all the items in the legend.

ChartLegend Property	Description
RowCount	Specifies the number of rows to be used in the legend.
ColumnCount	Specifies the number of columns to be used in the legend.
ItemsAlignment	Specifies the horizontal alignment of the items within the legend. Possible values: Near - Default value Center Far
ShowItemsShadow	Will render a shadow around the item image and text using the ItemsShadowColor. Default is false.
ItemsShadowColor	Specifies the color of the shadow to use. ShowItemsShadow should be set to true. Default is Gray.

ItemsShadowOffset	Specifies the breadth of the shadow. Default is {2, 2}.
ItemsSize	Specifies the size of the legend item rectangle. If the specified size is smaller than necessary to render the text, then it's ignored.
ItemsTextAlignment	Specifies the vertical alignment of the legend item text within the item bounds. Possible Values: Bottom Center - Default value Top
Spacing	Specifies the space between the legend borders and the legend items. Default is 4.
Text	Specifies the title text for the legend. You can set multiline text for the legend; Enter the text in the combobox and press ENTER key to begin a new line and CTRL+ENTER to set the entered multiline text.
TextColor	Specifies the color of the title text.
TextAlignment	Specifies the horizontal alignment of the title text. Possible Values: Center (Default value) Far Near

Table 4: ChartLegend Event

Event	Description	Arguments	Type	Reference links
MinSize	Used to specify a minimum rectangular size for the legend item.	object sender, ChartLegendMinSizeEventArgs e	NA	NA
DrawItem	Used to customize the rendering of the legend.	object sender, ChartLegendDrawItemEventArgs e	NA	NA

DrawItemText	Used to customize the rendering of the legend item text.	object sender, ChartLegendDrawItemT extEventArgs e	NA	NA
FilterItems	Used to dynamically provide a list of legend items during runtime.	object sender, ChartLegendFilterItemsE ventArgs e	NA	NA

ChartLegend Method	Description
GetItemBy	Gets the legend item at the specified coordinates.

You can also reference specific legend items and apply settings on them individually:

LegendItem Property	Description
BorderColor	Specifies the color of the border around the legend shape.
Font	Specifies the font for the text in this legend item.
Spacing	Specifies the space between this item and its adjacent items. Default is <b>20</b> .
Text	Specifies the text of the legend item. By default this will reflect the corresponding series name.
TextColor	Specifies the text color for this item.
IconAlignment	Specifies how the icon should be aligned within the item rectangle.
TextAlignment	Specifies how the text should be aligned within the item rectangle.
VisibleCheckBox	If this property is set to <b>true</b> , a checkbox will be shown beside the legend item through which the user can show/hide the corresponding series in the chart.
ShowShadow	Will render a shadow around the item image and text using the ItemsShadowColor.
ShadowOffset	Specifies the breadth of the shadow.

ShadowColor	Specifies the color of the shadow to use. ShowItemsShadow should be set to <b>true</b> . Default is <b>Gray</b> .
Children	Returns the child collection of the LegendItem.
IsChecked	Gets / sets the checkstate of the ChartLegendItem checkbox. By default it is set to <b>true</b> .
Visible	Lets you show / hide the legend item.

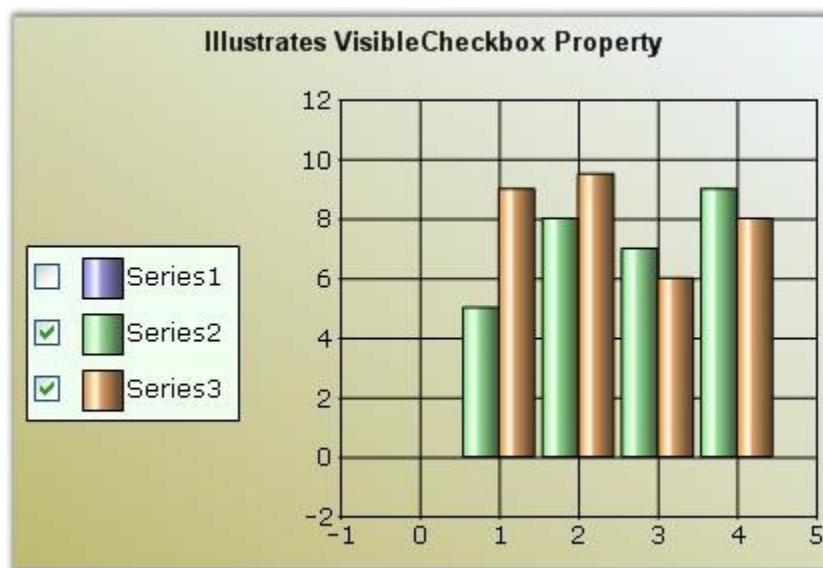


Figure 285: Chart with "Series1" Legend Item Unchecked

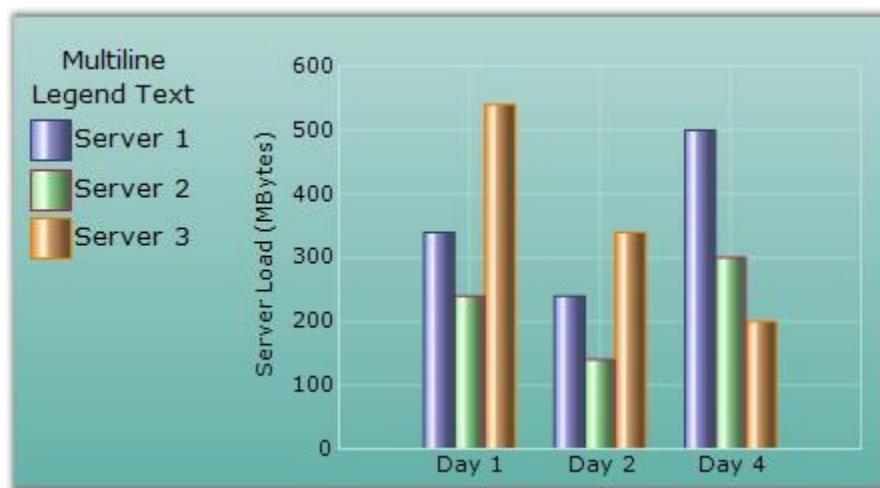


Figure 286: Chart with Multiline Legend Title 'Multiline Legend Text'

**See Also**

[ChartLegend](#), [Customizing LegendItem Image](#)

### 4.8.3 Customizing LegendItem Image

There are several options to customize the image rendered in the Legend. The following properties let you do so:

ChartLegend Property	Description
ShowSymbol	If <b>true</b> , the exact symbol rendered in the series data points will be used to render the icon in the legend as well. This overrides most of the other settings.
RepresentationType	<p>Specifies how each legend item should be represented, as the name implies:</p> <p><b>None (default setting)</b></p> <p><b>SeriesType</b> - An icon representing the series type.</p> <p><b>SeriesImage</b> - Will use the ImageList associated with the Series style.</p> <ul style="list-style-type: none"> <li>• Rectangle</li> <li>• Line</li> <li>• StraightLine</li> <li>• Circle</li> <li>• Diamond</li> <li>• Hexagon</li> <li>• Pentagon</li> <li>• Triangle</li> <li>• InvertedTriangle</li> <li>• Cross</li> </ul>

The following **ChartLegendItem** properties that can be accessed via the **Legend.Items** list typically override the above settings set in the Legend.

ChartLegendItem Property	Description
DrawSeriesIcon	Specifies if an icon representing the series type should be rendered for this legend item.
ImageList	Contains a collection of images and will be referred to, by the ImageList property.
ImageIndex	Specifies the index into the ImageList array which contains the image for this item.
Interior	Specifies the BrushInfo used to render the interior of a Chart Symbol.
RepresentationSize	Specifies the size of the rectangle inside which the associated image or symbol will get rendered.
ShowSymbol	If <b>true</b> , the exact symbol rendered in the corresponding series data points will be used to render the icon in this legend as well. This overrides most of the other settings.
Symbol	Symbols rendered in the Legend item can be customized using this property.
Type	<p>If ShowSymbol is <b>false</b>, you can customize the type of icon that gets rendered in the legend item. The default value will reflect the <b>ChartLegend.RepresentationType</b> setting.</p> <p>Possible Values:</p> <ul style="list-style-type: none"> <li>• Area</li> <li>• Circle</li> <li>• Cross</li> <li>• Diamond</li> <li>• Hexagon</li> <li>• Image</li> <li>• InvertedTriangle</li> <li>• Line</li> <li>• None</li> <li>• Pentagon</li> <li>• PieSlice</li> <li>• Rectangle</li> </ul>

	<ul style="list-style-type: none"><li>• Spline</li><li>• SplineArea</li><li>• StraightLine</li><li>• Rectangle</li></ul>
ShowIcon	If set to <b>false</b> , no icons will be rendered. This overrides most of the other settings including <b>ShowSymbol</b> .

### **Series Type Icon**

An icon representing the series type can be rendered in the legend.

To do this for all the legend items:

**[C#]**

```
this.chartControl1.Legend.RepresentationType =  
ChartLegendRepresentationType.SeriesType;
```

**[VB .NET]**

```
Me.chartControl1.Legend.RepresentationType =  
ChartLegendRepresentationType.SeriesType
```



*Figure 287: Legend Item with the Series Type Icon*

To do this for specific legend items,

**[C#]**

```
// The general setting affecting all Legend items could be anything  
this.chartControl1.Legend.RepresentationType =  
ChartLegendRepresentationType.SeriesImage;  
  
// This will force a specific legend item to show a series icon  
this.chartControl1.Legend.Items[0].DrawSeriesIcon = true;
```

**[VB.NET]**

```
'The general setting affecting all Legend items could be anything  
Me.chartControl1.Legend.RepresentationType =  
ChartLegendRepresentationType.SeriesImage  
  
'This will force a specific legend item to show a series icon  
Me.chartControl1.Legend.Items(0).DrawSeriesIcon = True
```

### **Series Symbol**

You can also choose to show the exact same symbol that is shown in the data points in a series.

To do this for all the legend items:

**[C#]**

```
//Set symbol for first series  
this.chartControl1.Series[0].Style.Symbol.Shape =  
ChartSymbolShape.Diamond;  
this.chartControl1.Series[0].Style.Symbol.Color = Color.Red ;  
this.chartControl1.Series[0].Style.Symbol.Size = new Size(7, 7);  
  
//This will cause the legend to render with the same symbol defined  
//above.  
this.chartControl1.Legend.ShowSymbol = true;  
  
//Setting RepresentationType to None to hide other representations  
this.chartControl1.Legend.RepresentationType =  
ChartLegendRepresentationType.None;
```

**[VB.NET]**

```
'Set symbol for first series  
Me.chartControl1.Series(0).Style.Symbol.Shape =  
ChartSymbolShape.Diamond  
Me.chartControl1.Series(0).Style.Symbol.Color = Color.Red  
Me.chartControl1.Series(10).Style.Symbol.Size = New Size(7, 7)  
  
'This will cause the legend to render with the same symbol defined  
//above.  
Me.chartControl1.Legend.ShowSymbol = True  
  
'Setting RepresentationType to None to hide other representations  
Me.chartControl1.Legend.RepresentationType =
```

```
ChartLegendRepresentationType.None
```

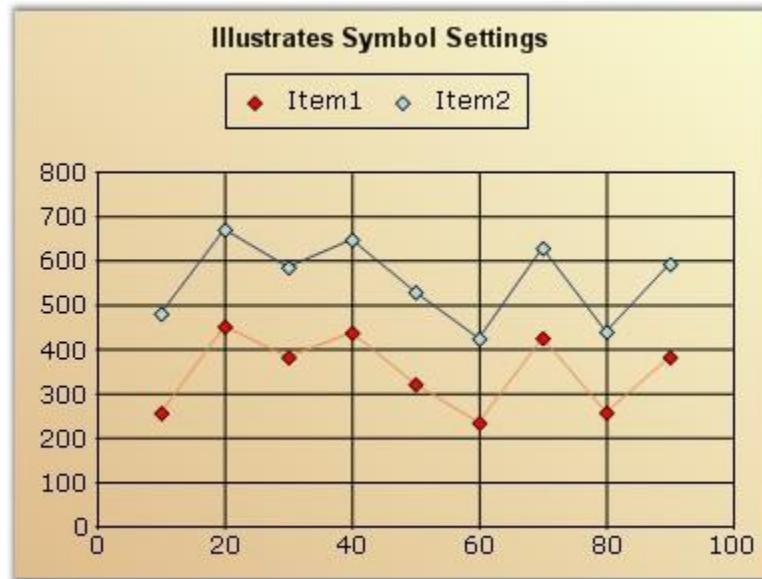


Figure 288: Legend Items rendered with the Same Symbol

### Custom Representation Icon

You can also choose to use one of the built-in representation icons in the legend items.

To do this for all the legend items:

[C#]

```
this.chartControl1.Legend.RepresentationType =
ChartLegendRepresentationType.Diamond;

// To specify a custom color for the interior of the icon
this.chartControl1.Legend.Items[0].Interior = new
BrushInfo(Color.Violet);
```

[VB.NET]

```
Me.chartControl1.Legend.RepresentationType =
ChartLegendRepresentationType.Diamond

'To specify a custom color for the interior of the icon
Me.chartControl1.Legend.Items(0).Interior = New BrushInfo(Color.Violet)
```



*Figure 289: Legend Item with a Custom Representation Icon*

To do the above only on specific legend items, use the **ChartLegendItem.Type** property.

### More Symbol Shapes

ChartLegendItem has the **Symbol** property, using which we can customize the symbols for particular legend items. This setting overrides the **Series[0].Style.Symbol** settings.

#### [C#]

```
//Series symbol settings
chartControl1.Legend.ShowSymbol = true;
chartControl1.Series[0].Style.Symbol.Shape = ChartSymbolShape.Diamond;
chartControl1.Series[0].Style.Symbol.Color = Color.AliceBlue;

//the above symbol settings is overridden by the following settings
chartControl1.Legend.Items[0].Symbol.Shape = ChartSymbolShape.Triangle;
chartControl1.Legend.Items[0].Symbol.Color = Color.Yellow;
```

#### [VB .NET]

```
'Series symbol settings
chartControl1.Legend.ShowSymbol = True
chartControl1.Series(0).Style.Symbol.Shape = ChartSymbolShape.Diamond
chartControl1.Series(0).Style.Symbol.Color = Color.AliceBlue

'the above symbol settings is overridden by the following settings
chartControl1.Legend.Items[0].Symbol.Shape = ChartSymbolShape.Triangle
chartControl1.Legend.Items[0].Symbol.Color = Color.Yellow
```

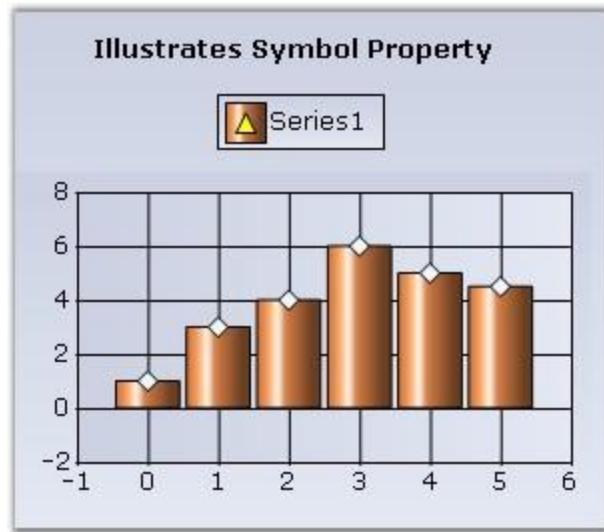


Figure 290: LegendItem customized with "Triangle" Symbol in "Yellow" Color

### Custom Images

You can also choose to show custom images in the legend items as follows:

#### [C#]

```
// Setting the representation type for the Legend items
this.chartControl1.Legend.RepresentationType =
ChartLegendRepresentationType.SeriesImage;
//Setting the image index
this.chartControl1.Legend.Items[0].ImageIndex = 0;
// The image will be picked up from this collection
series1.Style.Images = new
ChartImageCollection(this.imageList1.Images);

// Or from this collection, if available
this.chartControl1.Legend.Items[0].ImageList = new
ChartImageCollection(this.imageList1.Images);
```

#### [VB .NET]

```
'Setting the representation type for the Legend items
Me.chartControl1.Legend.RepresentationType =
ChartLegendRepresentationType.SeriesImage
'Setting the image index
Me.chartControl1.Legend.Items(0).ImageIndex = 0
```

```
' The image will be picked up from this collection
series1.Style.Images = New ChartImageCollection(Me.imageList1.Images)

' Or from this collection, if available
Me.chartControl1.Legend.Items(0).ImageList = New
ChartImageCollection(this.imageList1.Images)
```

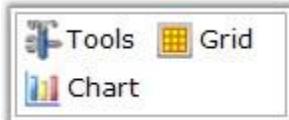


Figure 291: Chart Legend Items with Custom Images

### Hiding Icons

Icons for legend items can be hidden in any of the following ways:

#### [C#]

```
this.chartControl1.Legend.RepresentationType =
ChartLegendRepresentationType.None;

// To do this for a specific legend item:

// This will not even allocate any space for the icons
this.chartControl1.Legend.Items[0].ShowIcon = false;

// Or, set this. This will not render the icons, but will allocate some
empty space for it
this.chartControl1.Legend.Items[0].Type = ChartLegendItemType.None;
```

#### [VB.NET]

```
Me.chartControl1.Legend.RepresentationType =
ChartLegendRepresentationType.None

'To do this for a specific legend item

'This will not even allocate any space for the icons
Me.chartControl1.Legend.Items(0).ShowIcon = False

'Or, set this. This will not render the icons, but will allocate some
empty space for it
Me.chartControl1.Legend.Items(0).Type = ChartLegendItemType.None
```

## See Also

[ChartLegend](#), [ChartLegendItem](#)

## 4.9 Runtime Features

The following topics under this section elaborates on the runtime features of Chart control.

### 4.9.1 Zooming and Scrolling

#### Interactive Zooming

##### Zooming via Mouse

Essential Chart supports interactive zooming features along the x and y axis. During runtime, the user can simply select the range he wants to zoom with the mouse and the chart will accordingly zoom-in. Scrollbars will be activated to browse the areas that become hidden on zooming in.

Enable Zooming via the **EnableXZooming** and **EnableYZooming** properties.

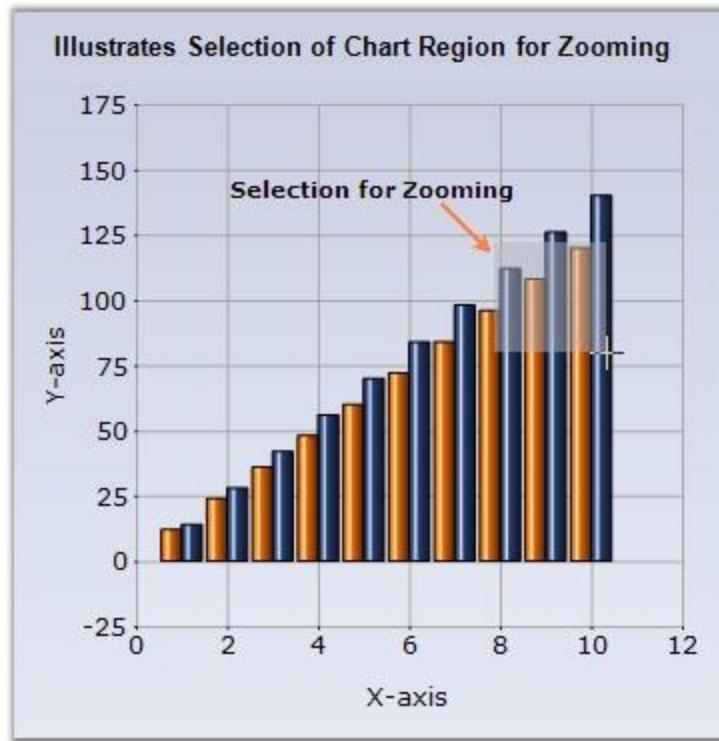
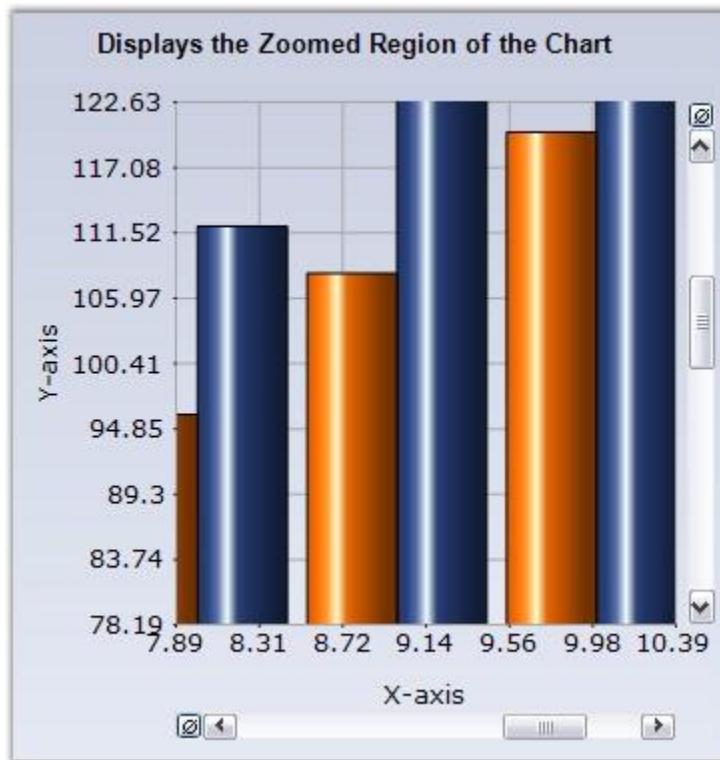


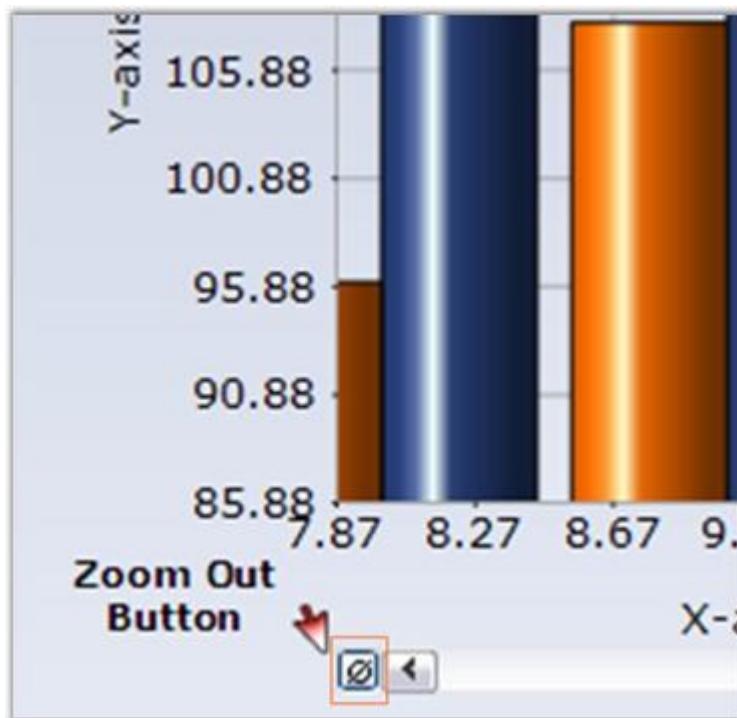
Figure 292: Select a region in the chart to Zoom-In



*Figure 293: Resultant Zoomed-In Chart*

The scrollbar will shift by the amount specified in the **ScrollPrecision** property which is set to 20 by default.

User can zoom out by clicking the "Zoom Out" button in the scrollbar.



*Figure 294: Zoom Out button beside the Scrollbar*

**ZoomOutIncrement** property specifies the increment by which to zoom out. The default value is 0.2.

### **Programmatic Zooming**

Programmatically the chart can be zoomed using **ZoomFactorX** and **ZoomFactorY** properties. The Zoom factor is usually between 0 and 1. When set to 1, the chart isn't zoomed. When set to 0.5, the chart is double its usual size. Scrollbars will automatically appear to allow any section of the hidden range to be viewed. The default value is 1.

You can also programmatically specify the scrollbar position of the zoomed in axes using the **ZoomPositionX** and **ZoomPositionY** properties.

To restrict the zoom-in factor to a certain level on the x and y axis use the **MinZoomFactorX** and **MinZoomFactorY** properties. The value can be in between 0 and 1. 1 means not zoomed.

### Zooming via Keyboard

Essential Chart also enables users to use keyboard shortcuts to enable zooming. Enable this feature through the **KeyZoom** property.

Using the following properties the zooming action can be mapped to specific keys.

Chart control Property	Description
ZoomCancel	Specifies the keyboard shortcut to control Zoom cancel. The default value is ESCAPE.
ZoomDown	Specifies the keyboard shortcut to control Zoom Down. The default value is DOWN arrow.
ZoomIn	Specifies the keyboard shortcut to control Zoom In. The default value is ADD key.
ZoomLeft	Specifies the keyboard shortcut to control Zoom Left. The default value is LEFT arrow.
ZoomOut	Specifies the keyboard shortcut to control Zoom Out. The default value is SUBTRACT.
ZoomRight	Specifies the keyboard shortcut to control Zoom Right. The default value is RIGHT arrow.
ZoomUp	Specifies the keyboard short cut to control Zoom Up. The default value is UP arrow.

### Panning Support for Zoomed Chart

Now, you will be able to pan a chart when it is zoomed. Set the **ChartControl.MouseAction** to 'Panning' to enable this feature. Set the MouseAction to 'None' to disable this feature. The panning action can be controlled using the **ZoomActions** property that is available for individual axis.

Chart Axes Property	Description
ZoomActions	Specifies the zoom action on the corresponding axis. The options are,

	<p><i>Panning</i> - Enables panning in the zoomed chart. <i>None</i> - Disables panning in the zoomed chart.</p>
--	----------------------------------------------------------------------------------------------------------------------

**[C#]**

```
this.chartControl1.MouseAction = ChartMouseAction.Panning;
this.chartControl1.PrimaryXAxis.ZoomActions =
ChartZoomingAction.Panning;
this.chartControl1.PrimaryYAxis.ZoomActions =
ChartZoomingAction.Panning;
```

**[VB .NET]**

```
Me.chartControl1.MouseAction = ChartMouseAction.Panning
Me.chartControl1.PrimaryXAxis.ZoomActions = ChartZoomingAction.Panning
Me.chartControl1.PrimaryYAxis.ZoomActions = ChartZoomingAction.Panning
```



**Note:** Remember to enable zooming on both the axis using *EnableXZooming* and *EnableYZooming* properties, before trying out the above panning feature. You cannot pan a chart without zooming it.

### Formatted Axes Labels

It is possible to show formatted axes labels for a zoomed chart. Essential Chart's **SmartDateZoom** property when set to **true** enables this feature. You can set any one of the following custom label formats to the chart axis.

- SmartDateZoomDayLevelLabelFormat
- SmartDateZoomYearLevelLabelFormat
- SmartDateZoomWeekLevelLabelFormat
- SmartDateZoomSecondLevelLabelFormat
- SmartDateZoomMonthLevelLabelFormat
- SmartDateZoomHourLevelLabelFormat
- SmartDateZoomMinuteLevelLabelFormat

**[C#]**

```
this.chartControl1.PrimaryXAxis.SmartDateZoom = true;
this.chartControl1.PrimaryXAxis.SmartDateZoomDayLevelLabelFormat = "dd
MM/yy HH:00";
```

**[VB .NET]**

```
Me.chartControll.PrimaryXAxis.SmartDateZoom = True  
Me.chartControll.PrimaryXAxis.SmartDateZoomDayLevelLabelFormat = "dd  
MM/yy HH.00"
```

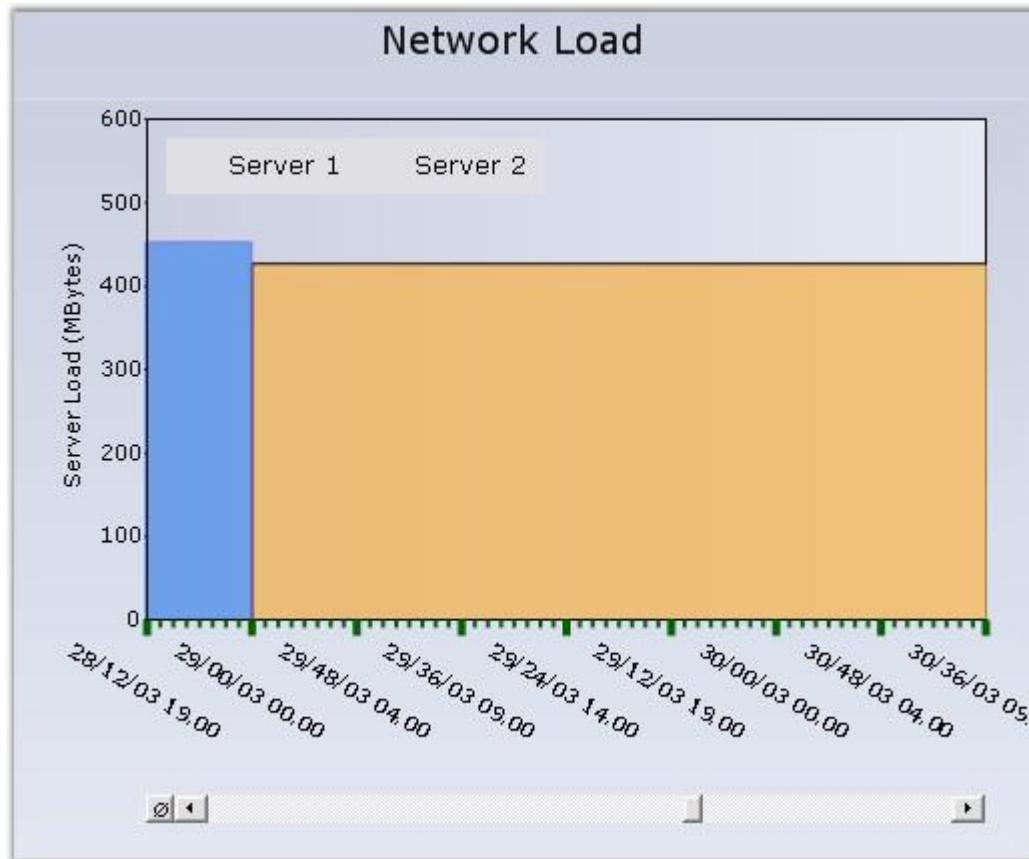


Figure 295: SmartDateZoomDayLevelLabelFormat = "dd MM/yy HH.00"



**Note:** The value type of the axis should be "DateTime" for setting the above formatted labels.

A sample which demonstrates the zooming and scrolling features is available in the following sample installation location.

<Install Location>\Syncfusion\EssentialStudio\<Install  
version>\Web\chart.web\Samples\3.5\UserInteraction\ZoomingAndScrolling

## See Also

[How to hide the Chart ZoomButton](#)

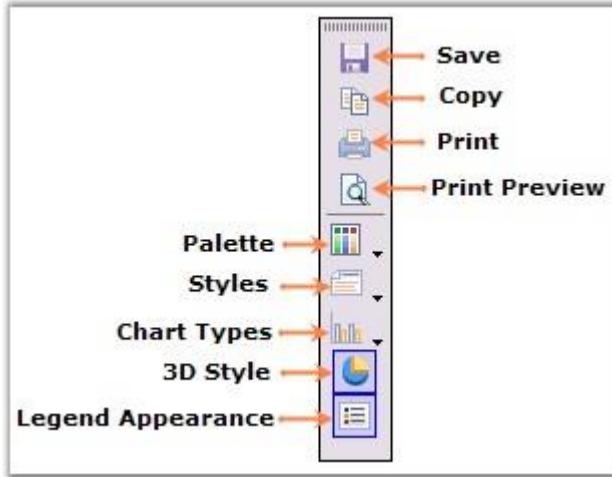
## 4.9.2 Toolbars

Essential Charts comes with a built-in Toolbar that can be made visible to enable the user to do the following during runtime.

- Save the chart as an image.
- Copy the image to clipboard.
- Print the chart.
- Print Preview of the Chart.
- Change the color palette of the chart.
- Affects the style of the chart.
- Change the Chart Type.
- Toggle 3D style of the Chart.
- Toggle Legend Appearance.

The toolbar can be made visible through the ChartControl's **ShowToolbar** property.

The toolbar looks like the below image.



*Figure 296: Built-In Chart Toolbar*

The toolbar commands and their functionalities are described below.

Chart toolbar Commands	Chart toolbar Items name	Description
------------------------	--------------------------	-------------

Save	ChartToolbarSaveltem	Using this command, user can save the chart to a specific location.
Copy	ChartToolBarCopyItem	Clicking this toolbar command will copy the chart to the clipboard.
Styles	ChartToolBarStyleItem	This popup a Chart Series Style dialog window, using which various properties and chart styles can be set.
Print	ChartToolBarPrintItem	This toolbar command is used to print the Chart.
Palette	ChartToolBarPalettesItem	Palette for the series can be chosen at run time using this command. All palette colors available in the designer will be available in this Palette option also.
Chart Types	ChartToolBarTypeItem	Any chart type can be set for the chart at run time using this command.
Print Preview	ChartToolBarPrintPreviewItem	This toolbar command is used to see a print preview of the Chart.
Toggling 3D	ChartToolBarSeries3DItem	This command is used to toggle the 3D mode of the chart.
Toggle Legend Appearance	ChartToolBarShowLegendItem	This command is used to toggle the legend appearance.
Splitter	ChartToolBarSplitter	This item provides a logical split between the collection of commands.

### Custom Toolbar Commands

You can also add custom toolbar items using **ChartToolBarCommandItem** class. The **ChartCommands** enum lists the commands that can be added. The following table describes those commands.

Chart toolbar Custom Commands	Description
ZoomIn	Using this command, user can zoom the chart.
ZoomOut	This command zooms out the chart.
ResetZooming	The zooming is reset using this command.
AutoHighlight	This command is used to enable the autohighlight feature in the chart series.
ToggleXZooming	This toolbar command enables zooming in x-axis.
ToggleYZooming	This toolbar command enables zooming in y-axis.
TogglePanning	This command enables panning of the zoomed chart.

**[C#]**

```
ChartToolBarCommandItem x1 = new ChartToolBarCommandItem();
x1.Command = ChartCommands.AutoHighlight;
x1.IsCheckedable = false;
Image v = System.Drawing.Image.FromFile(@"..\..\Data\Visio.png");
x1.Image = v;
x1.Name = "Custom Tools";
x1.ToolTip = "Highlighting";
x1.Checked = true;
this.chartControl1.ToolBar.Items.Add(x1);
```

**[VB.NET]**

```
Dim x1 As New ChartToolBarCommandItem()
x1.Command = ChartCommands.AutoHighlight
x1.IsCheckedable = False
Dim v As Image = System.Drawing.Image.FromFile("../..\Data\Visio.png")
x1.Image = v
x1.Name = "Custom Tools"
x1.ToolTip = "Highlighting"
x1.Checked = True
Me.chartControl1.ToolBar.Items.Add(x1)
```



Figure 297: *CustomCommand = "ChartCommands.AutoHighlight" ; Command ToolTip = "Highlighting"*



Figure 298: *AutoHighlight feature enabled in Chart using Custom Toolbar Command*

#### 4.9.2.1 Toolbar Properties

The chart control provides complete support for customizing the toolbar appearance. Use the **ChartControl.ToolBar** property to access the toolbar. At runtime, double-click the toolbar to show the **ToolBar Properties** dialog box as in the below image, which lists all the properties. For this, you need to set the **ToolBar.ShowDialog** property to **True**. If you do not want to display this dialog box, set this property to **False**.

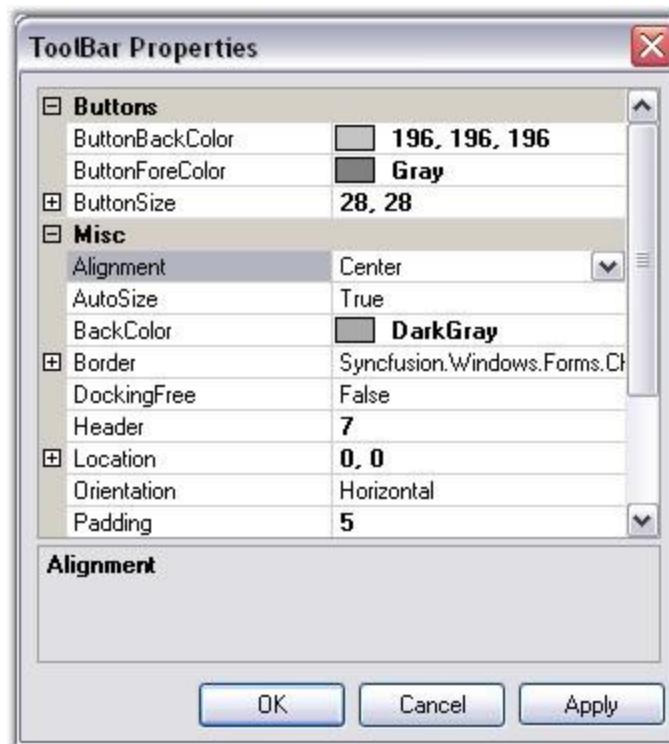


Figure 299: ToolBar Properties Dialog Box

Below are the toolbar properties and their description.

Chart ToolBar Property	Description
Alignment	Indicates the alignment of the toolbar. Default value is <b>Center</b> .
AutoSize	Indicates if the toolbar can be resized automatically. Default value is <b>true</b> .
BackColor	Indicates back color of the toolbar.
Border	Specifies the border style.
Buttons	List of buttons to which you can add new Buttons or delete existing ones.
ButtonBackColor	Gets / sets the back color of the toolbar button.
ButtonFlatStyle	Gets / sets the flat style appearance for the toolbar button control. Default value is <b>FlatStyle.Flat</b> .
ButtonForeColor	Gets / sets the fore color of the toolbar button.
ButtonSize	Indicates the button size of the toolbar buttons.

DockingFree	Indicates if the toolbar is to be held docked. Default value is <b>false</b> .
Header	Gets / sets the height of the header. Default value <b>0</b> .
Location	Gets / sets the location of the toolbar.
Orientation	Gets / sets the orientation of the toolbar. Default value is <b>Horizontal</b> .
Position	Gets / sets the docking position of the toolbar. Default value is <b>ChartDock.Top</b> .
ShowBorder	Indicates if the border of the toolbar should be shown. Default value is <b>true</b> .
Size	Gets / sets the size of the toolbar button. Will be used only when Autosize property is set to <b>false</b> .
Spacing	Gets or sets the spacing. Default value is <b>4</b> .

#### 4.9.2.2 Appearance

##### Setting Styles for the Chart through the Toolbar

Click the **Styles** icon in the toolbar to open the **Chart Series Style** dialog box. The following are the settings available in this dialog box.

- Interior color for the series can be set using the options available in the **Interior** tab.
- Border properties using **Border** tab.
- Text for the series can be enabled and also customized using the **Text** tab.
- Shadow for the series can be enabled and customized using the **Shadow** tab.
- Series can hold customized symbols using the **Symbol** tab.
- FancyToolTip can be enabled using the options available in the **Fancy ToolTip** tab.

The below image shows how to set the interior properties through "Interior" tab in the Chart Series Style Window. This can be invoked by clicking "Styles" command.

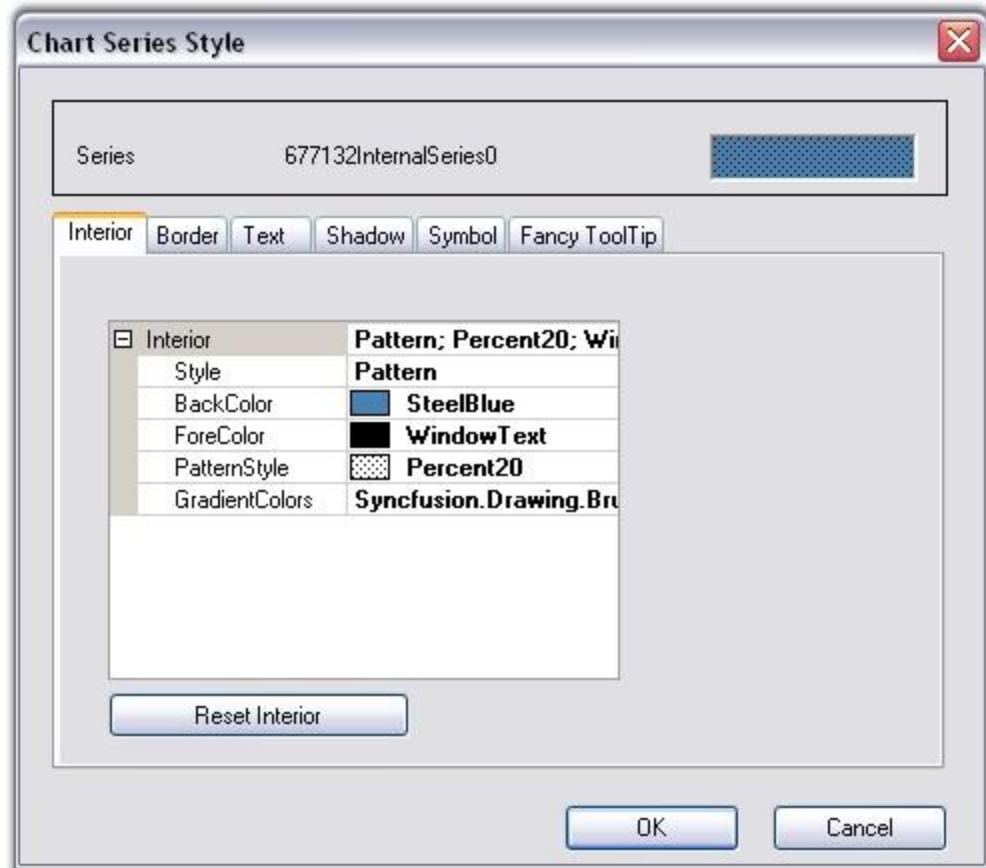


Figure 300: Chart Series Style Window to set Interior Properties

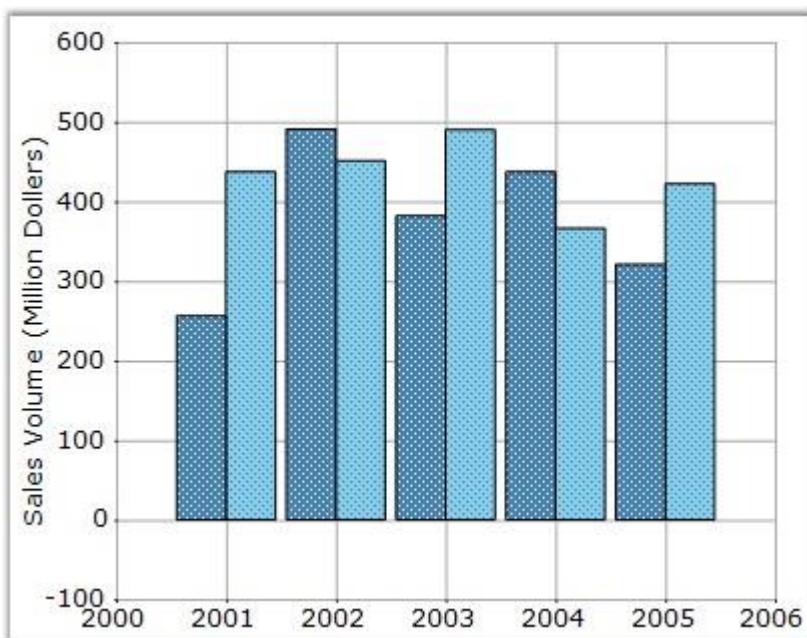


Figure 301: Chart after setting Interior Properties

### Toolbar Appearance

Toolbar provides an option to set different back color, border style, button back color and button fore color.

User can enable or disable the Border line of Toolbar by using **ShowBorder** property in the Toolbar instance.

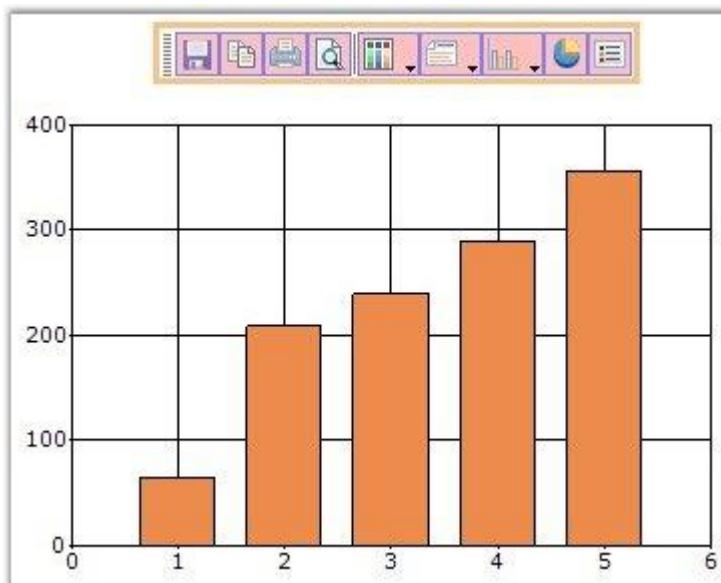


Figure 302: Toolbar with Border

### Toolbar Behavior

The docking behavior of the Toolbar can be controlled using **Toolbar.Behavior** property.

Toolbar Property	Description
Behavior	<p>Specifies the docking behavior of the toolbar.</p> <p>Docking - It is dockable on all four sides.</p> <p>Movable - It is movable.</p> <p>All - It is movable and dockable.</p> <p>None - It is neither movable nor dockable.</p>

**[C#]**

```
this.chartControl1.ToolBar.Behavior = ChartDockingFlags.All;
```

**[VB .NET]**

```
Me.chartControl1.ToolBar.Behavior = ChartDockingFlags.All
```



**Note:** You can display or hide a toolbar while printing a Chart. See [Printing And Print Preview topic](#) for more details.

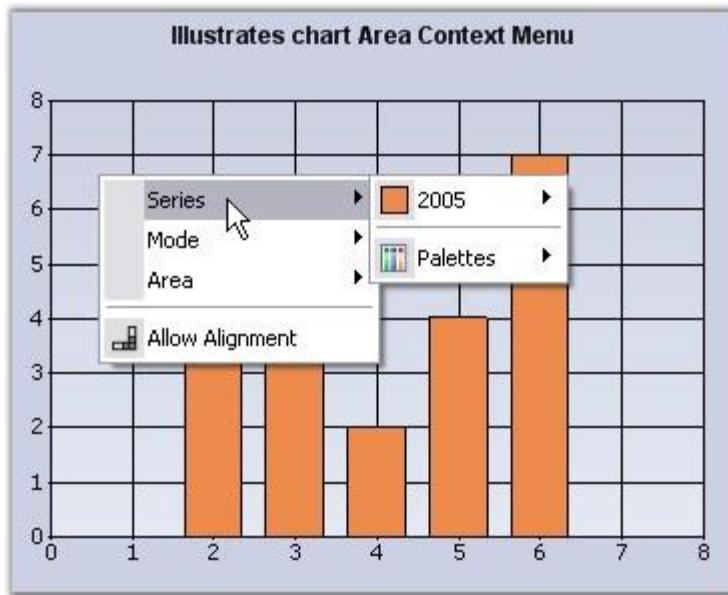
### 4.9.3 Context Menu

#### Chart Area and Series Context menu

The chart has a built-in context menu, which can be enabled by setting the **ShowContextMenu** property to **true**. This context menu will let the user change the chart type on a series, enable zooming, switch between 2D and 3D modes and so on.

There are two types of context menus, both of which get shown by default when the above property is set to **true**.

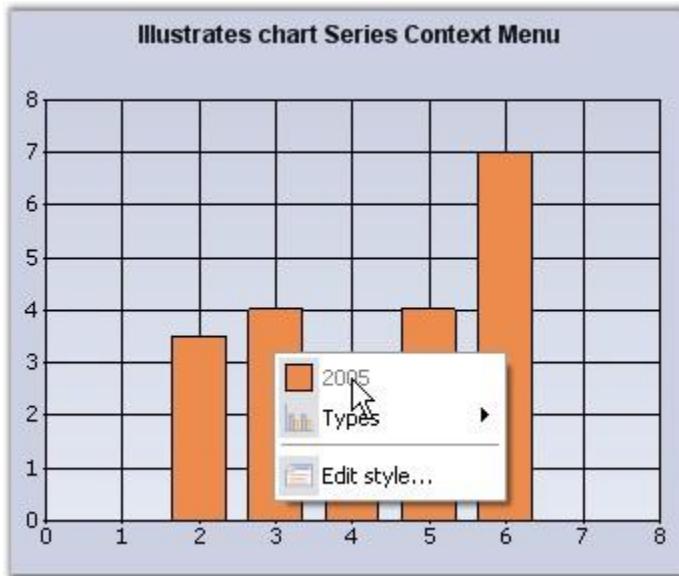
1. Chart Area context menu - This will be displayed when the mouse is over the chart area.



*Figure 303: Chart Area Context Menu*

This context menu can be disabled by setting the **DisplayChartContextMenu** property to **false**.

2. Chart Series context menu - This will be displayed when the mouse is over a series.

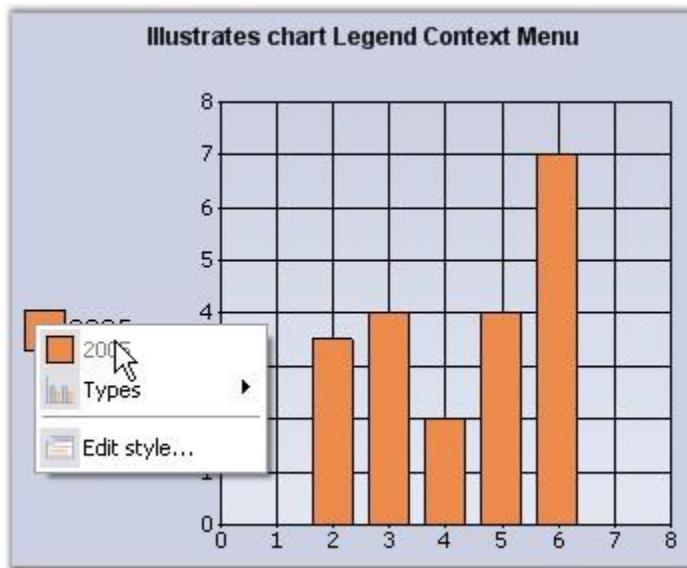


*Figure 304: Chart Series Context Menu*

This context menu can be disabled by setting the **DisplaySeriesContextMenu** property to **false**.

### Legend Context Menu

This context menu can be enabled by setting the **ShowContextMenuInLegend** property to **true**.



*Figure 305: Legend Context Menu*

### 4.9.4 Interactive Features

#### Interactive Cursor

This feature lets you position the mouse pointer at a specific data point in a series and hint you on its x and y values via a horizontal and vertical line passing through the data point and intersecting the x and y axis. These lines can be dragged around in order to position them at specific data points.

Interactive Cursor can be implemented by creating an instance of **ChartInteractiveCursor** with the ChartSeries as its input. Then add the instance to the Interactive Cursors collection as shown below.

**[C#]**

```
// Create a new instance of the ChartInteractiveCursor class and
// initialize chartseries into it.
ChartInteractiveCursor cursor1 = new
ChartInteractiveCursor(this.chartControl1.Series[0]);

// Add the instance to the ChartInteractive Cursor collection.
this.chartControl1.ChartArea.InteractiveCursors.Add(cursor1);

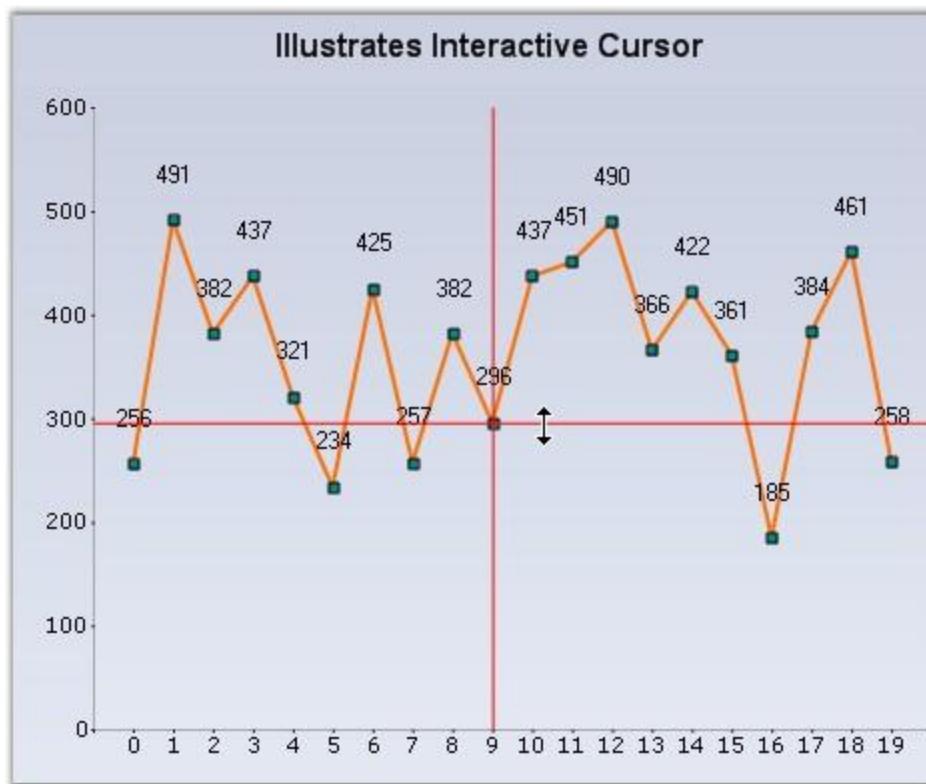
//Color of the pointer
cursor1.Color = Color.Red;
```

**[VB .NET]**

```
' Create a new instance of the ChartInteractiveCursor class and
' initialize chartseries into it.
ChartInteractiveCursor cursor1 = New
ChartInteractiveCursor(Me.chartControl1.Series(0))

' Add the instance to the ChartInteractive Cursor collection.
Me.chartControl1.ChartArea.InteractiveCursors.Add(cursor1)

'Color of the pointer
cursor1.Color = Color.Red
```



*Figure 306: Chart Interactive Cursor*

#### **Chart AutoHighlight**

The points or the series of the chart can be highlighted when the mouse hovers over them. Use the **AutoHighlight** property to enable this feature.

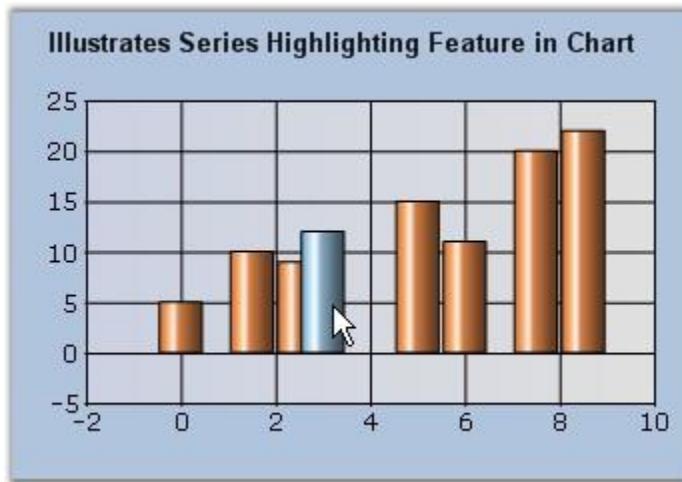


Figure 307: Column Highlighted on Mouse Hover

## Chart Series Highlighting

You can also highlight a particular chart series alone while mouse hovering, and make the other series transparent. For this, you need to set **SeriesHighlight** property to **true**. The series can also be highlighted by hovering the mouse over a legend item corresponding to a particular series.

The following table describes properties related to this feature.

Property	Description
HighlightInterior	Sets the highlight color for the series.
HiddenInterior	Controls the transparency of the non-highlighted series. While mouse hovering on a particular series, all other series will be set with the color, specified in this property.
SeriesHighlightIndex	If you want to highlight only a particular series alone, you need to set the index value for this property. The default value is <b>-1</b> .



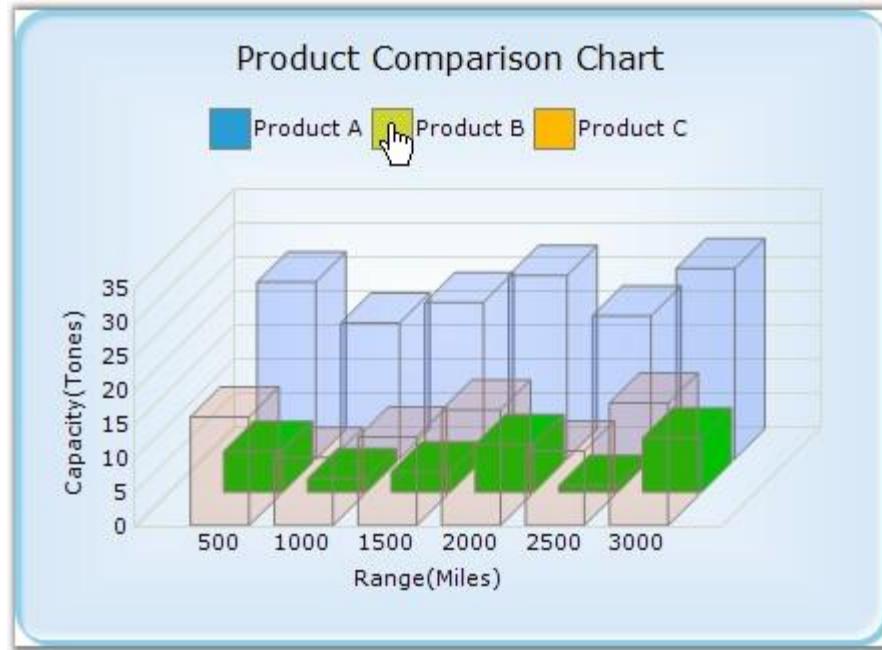
**Note:** The **AutoHighlight** property should be disabled to enable this chart series highlighting feature.

### [C#]

```
this.chartControl1.SeriesHighlight = true;
this.chartControl1.Series[0].Style.HighlightInterior = new
BrushInfo(Color.Gold);
BrushInfo bi = new BrushInfo(GradientStyle.Vertical, Color.Red,
Color.Red);
this.chartControl1.Series[0].Style.HiddenInterior = new BrushInfo(0,
bi);
```

### [VB .NET]

```
Me.chartControl1.SeriesHighlight = True
Me.chartControl1.Series(0).Style.HighlightInterior = New
BrushInfo(Color.Gold)
Dim bi As New BrushInfo(GradientStyle.Vertical, Color.Red, Color.Red)
Me.chartControl1.Series(0).Style.HiddenInterior = New BrushInfo(0, bi)
```



*Figure 308: Product B Series being Highlighted*

#### **4.9.4.1 Drawing Interactive Cursor Separately – Either Horizontally or Vertically or Both**

An Interactive cursor is used to indicate the x-axis and y-axis values of a data point. The interactive cursor can be drawn in different orientations namely Horizontal, Vertical and in both directions. The cursor color can also be changed according to requirements. The default color is set at the initial stage and this can be changed according to orientation or a common color can be set for both orientations as the parent color.

##### **Use Case Scenarios**

The purpose of using Chart Interactive Cursor is to indicate the x-axis and y-axis values for a specified data point. You can accurately locate the position of the point on the axes. You can use it as per your requirement i.e. horizontal, vertical or both accordingly.

The following screen shot shows the Interactive cursor, which is drawn in horizontal orientation:

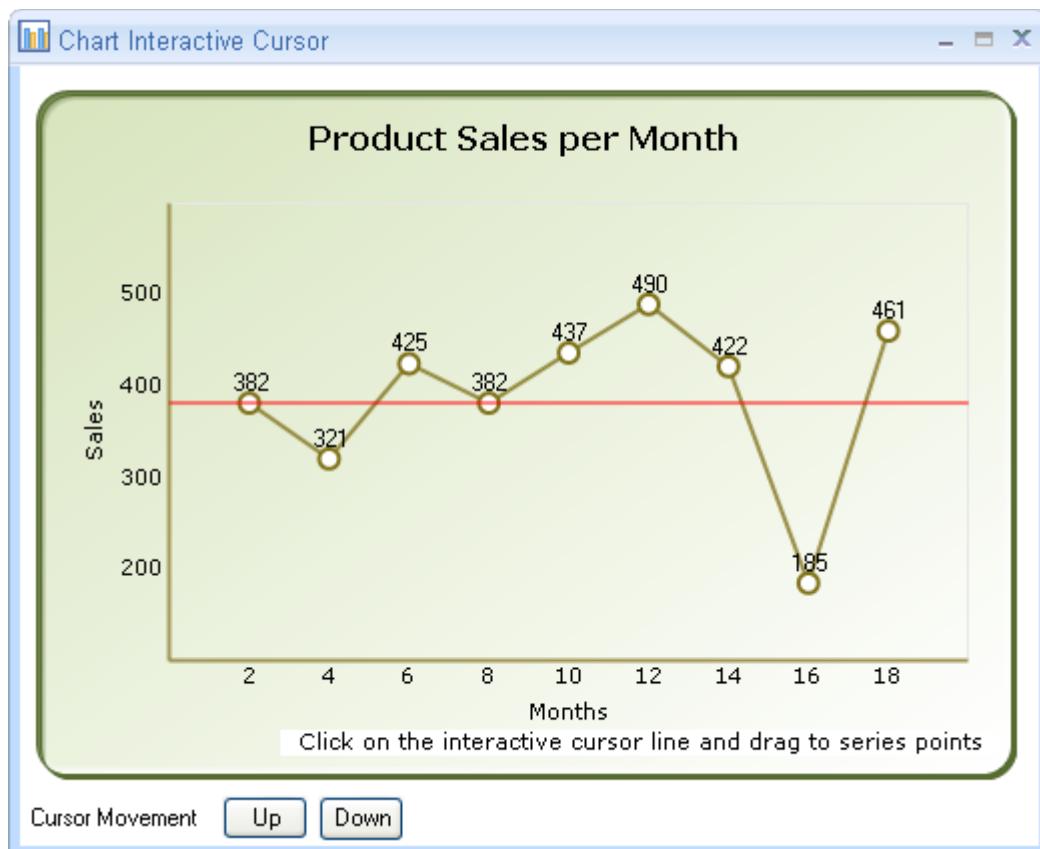


Figure 309: Interactive Cursor with Horizontal Orientation

## Properties

Property	Description	Data Type
CursorOrientation	Indicates the orientation in which the Interactive Cursor is to be drawn. The options are : <ul style="list-style-type: none"> <li>• Horizontal</li> <li>• Vertical</li> <li>• Both</li> </ul>	Enum
HorizontalCursorPosition	Specifies the color, which is to be used when Horizontal Interactive Cursor is drawn	Color
VerticalCursorPosition	Specifies the color, which is to be used when Vertical	Color

	Interactive Cursor is drawn	
Color	Specifies the base color, which is to be used other than the default color. This acts as a parent color.	Color

### Drawing Interactive Cursor in a Chart Application

To add Interactive Cursor to the Chart control:

1. Add a Interactive cursor to the **Chart** control.
2. Set the orientation to horizontal or vertical or both.
3. Choose the color.

Refer to the following code snippets to draw the interactive cursor separately.

[C#.NET]

```
cursor1 = new ChartInteractiveCursor(this.chartControl1.Series[0]);
this.chartControl1.ChartArea.InteractiveCursors.Add(cursor1);
cursor1.CursorOrientation = InteractiveCursorOrientation.Horizontal;
cursor1.HorizontalCursorColor = Color.Red;
```

**[VB .NET]**

```
cursor1 = New ChartInteractiveCursor(Me.chartControl1.Series(0))  
Me.chartControl1.ChartArea.InteractiveCursors.Add(cursor1)  
cursor1.CursorOrientation =  
InteractiveCursorOrientation.Horizontal  
cursor1.HorizontalCursorColor = Color.Red
```

The interactive cursor as described earlier can be set in three different orientations.

To draw the interactive cursor in horizontal orientation, you need to set the cursor orientation to “Horizontal” as shown in the following code snippets:

**[C# .NET]**

```
cursor1.CursorOrientation =  
InteractiveCursorOrientation.Horizontal;
```

**[VB .NET]**

```
cursor1.CursorOrientation =  
InteractiveCursorOrientation.Horizontal
```

The same step is repeated for “vertical” and “both” cursor orientations except for the naming “Vertical” and “Both” respectively.

You can also add color(s) to individual interactive cursor. The default color (base color) is Red. You can change the default color by using Color, HorizontalCursorColor, and VerticalCursorColor properties. When you use the Color property, the interactive cursor will be drawn based on the color specified by the Color property (assuming this as base/parent color) regardless of the colors specified for Horizontal and Vertical cursor orientations. This is shown in the following code snippets:

**[C# .NET]**

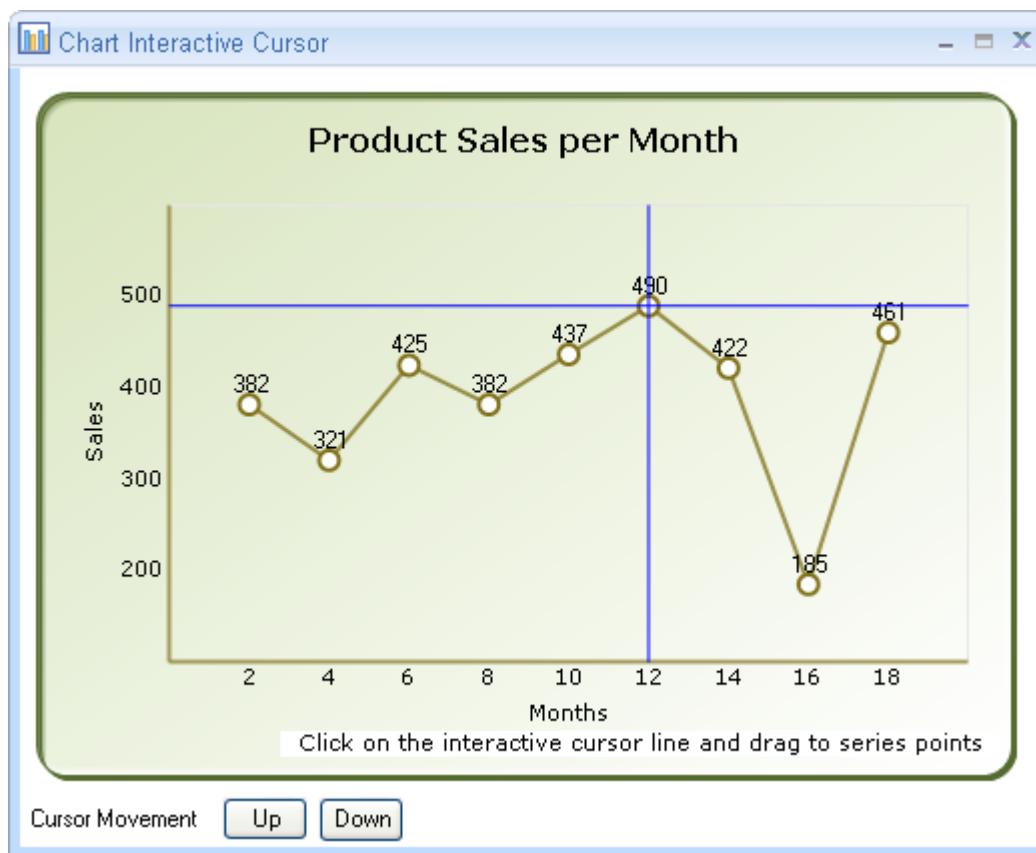
```
cursor1.CursorOrientation = InteractiveCursorOrientation.Both ;  
cursor1.Color = Color.Blue;
```

```
cursor1.VerticalCursorColor = Color.Green;  
cursor1.HorizontalCursorColor = Color.Red;
```

**[VB.NET]**

```
cursor1.CursorOrientation = InteractiveCursorOrientation.Both  
cursor1.Color = Color.Blue  
cursor1.VerticalCursorColor = Color.Green  
cursor1.HorizontalCursorColor = Color.Red
```

Now, the default color would be replaced with blue color at both the orientations as it is the parent color.



*Figure 310: Interactive Cursor with Parent Color Set to Blue*

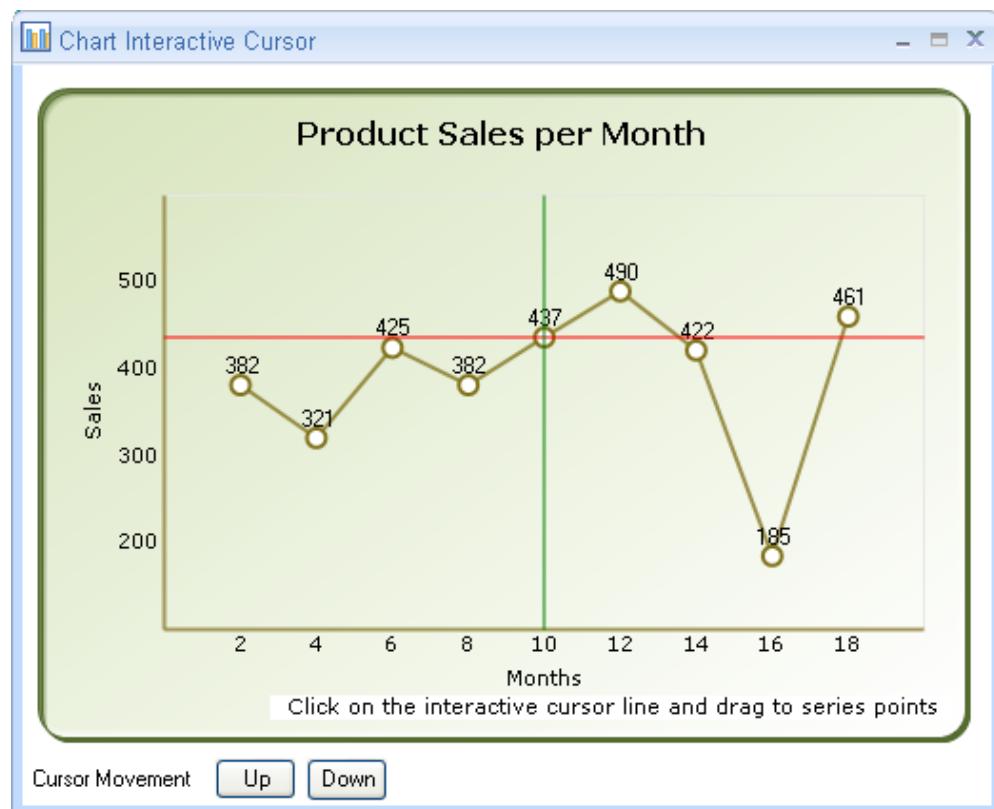
The following code snippets draw interactive cursor in different colors:

[C# .NET]

```
cursor1.CursorOrientation = InteractiveCursorOrientation.Both ;  
cursor1.VerticalCursorColor = Color.Green;  
cursor1.HorizontalCursorColor = Color.Red;
```

[VB .NET]

```
cursor1.CursorOrientation = InteractiveCursorOrientation.Both  
cursor1.VerticalCursorColor = Color.Green  
cursor1.HorizontalCursorColor = Color.Red
```



*Figure 311 : Interactive Cursor with Horizontal Cursor Color Red and Vertical Cursor Color Green.*

#### 4.9.4.2 ChartInteractiveCursor Support for Chart Area

Essential Chart now supports moving the interactive cursor fully over the chart area. It provides simple methods to display symbols at the intersection of series points and the interactive cursor.

##### Use Case Scenarios

This feature is useful for moving the interactive cursor across the entire chart area region, allowing users to get the intersection point values between the series and interactive cursor.

##### Sample Link

To view a sample,

1. Open the **Syncfusion Dashboard**.
2. Click the **Windows Forms** drop-down list and select **Run Locally Installed Samples**.
3. Navigate to **Chart samples > User Interaction > Chart Interactive Cursor**.

##### Properties

Property	Description	Type	Data Type
MoveToChartArea	Specifies whether the interactive cursor is enabled for chart series or series points	Server Side	Boolean
XInterval	Specifies the cursor movement on the x-axis (left to right or right to left)	Server Side	Double
YInterval	Specifies the cursor movement on the y-axis (top to bottom or bottom to top)	Server Side	Double

##### Methods

Method	Description	Parameters	Type	Return Type
SetSeriesSymbolForCursor	Used to customize the symbol which will be displayed at the intersection between the series point and interactive cursor.	ChartSymbolInfo	Server Side	Void

## Existing Features

We can move the interactive cursor for series points only (i.e., the interactive cursor can be moved from one data point to another by dragging). Users cannot move the interactive cursor over the whole chart area.

### MoveToChartArea

We can enable this feature by setting the **MoveToChartArea** property of the interactive cursor to **true**. The default value is **false**.

#### [C#]

```
this.chartControl1.ChartArea.InteractiveCursors[0].MoveToChartArea =  
true;
```

#### [VB]

```
Me.chartControl1.ChartArea.InteractiveCursors(0).MoveToChartArea = True
```

### XInterval

The cursor on the x-axis can be moved from left to right or right to left based on the value provided in this property of the interactive cursor.

### YInterval

The cursor on the y-axis can be moved from top to bottom or bottom to top based on the value provided in this property of the interactive cursor.

### Symbol

Symbols will be displayed when the interactive cursor meets the series point in the chart area by dragging.

#### [C#]

```
this.chartControl1.ChartArea.InteractiveCursors[0].XInterval = 2;  
this.chartControl1.ChartArea.InteractiveCursors[0].YInterval = 50;
```

#### [VB]

```
Me.chartControl1.ChartArea.InteractiveCursors(0).XInterval = 2  
Me.chartControl1.ChartArea.InteractiveCursors(0).YInterval = 50
```



Figure 312: Symbols shown at intersection between series point and interactive cursor

## 4.9.5 ToolTips

Essential Chart supports ToolTips in different areas of the chart which comes with multiple customization options.

The different tooltips in the chart can be turned off using the control's **ShowToolTips** property.



**Note:** The **ShowToolTips** property in the chart is false by default, so remember to turn this on, before setting tooltips in the different chart areas.

### DataPoint Tooltips

Tooltips can be shown on each data point when the mouse hovers on them. The format of the tooltip is specified by the following property in ChartSeries.

ChartSeries Property	Description
PointsToolTipFormat	Specifies the format for the datapoint tooltips. The following place-holders can be used in the value.

- |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>{0} - Will be replaced by the corresponding <b>ChartSeries.Name</b>.</p> <p>{1} - Will be replaced by the corresponding <b>ChartSeries.Style.ToolTip</b>.</p> <p>{2} - Will be replaced by the corresponding data point's tooltip, for example to set the first point's tooltip, use "series1.Styles[0].ToolTip".</p> <p>{3} - Will be replaced by the corresponding X value of the point.</p> <p><b>{4}</b> - Will be replaced by the corresponding Y value of the point. <b>Default setting</b>.</p> <p>{5} - Will be replaced by the 2nd Y value, if any.</p> <p>{6} - and so on.</p> |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

[C#]

```
series1.PointsToolTipFormat = "Sales:{4}K";
```

[VB .NET]

```
series1.PointsToolTipFormat = "Sales:{4}K"
```

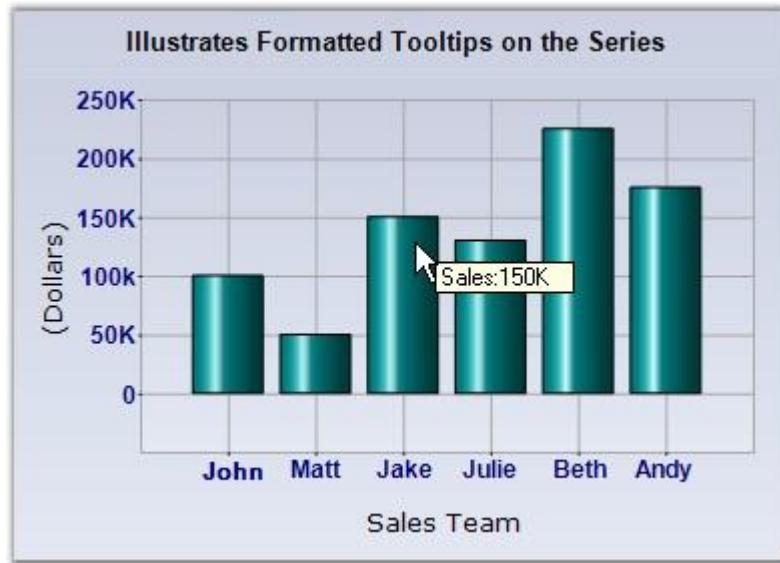


Figure 313: Tooltip Format set for Data Points

You can also customize the tooltip for individual data points by setting the **ToolTip** style for each data point. This is best accomplished by listening to the **ChartSeries.PrepareStyle** event as shown below.

**[C#]**

```
//Setting the Tooltip Format
series1.PointsToolTipFormat = "{2}";

protected void series1_PreparesStyle(object sender,
ChartPrepareStyleEventArgs args)
{
    // Style formatting using a callback. You can apply the same settings
    // directly on the series style on the
    // point styles.
    ChartSeries series = sender as ChartSeries;
    if (series != null)
    {
        args.Style.ToolTip = "Made " + (
(series.Points[args.Index].YValues[0] / 150) * 100) + "% of quota";
        args.Handled = true;
    }
}
```

**[VB.NET]**

```
'Setting the Tooltip Format
series1.PointsToolTipFormat = "{2}"

Protected Sub series1_PreparesStyle(ByVal sender As Object, ByVal args
As ChartPrepareStyleEventArgs)
    ' Style formatting using a callback. You can apply the same settings
    // directly on the series style on the
    ' point styles.
    Dim series As ChartSeries = CType(If(sender Is ChartSeries,
sender, Nothing), ChartSeries)
    If Not series Is Nothing Then
        args.Style.ToolTip = "Made " + (
(series.Points(args.Index).YValues[0] / 150) * 100) + "% of quota"
        args.Handled = True
    End If
End Sub
```

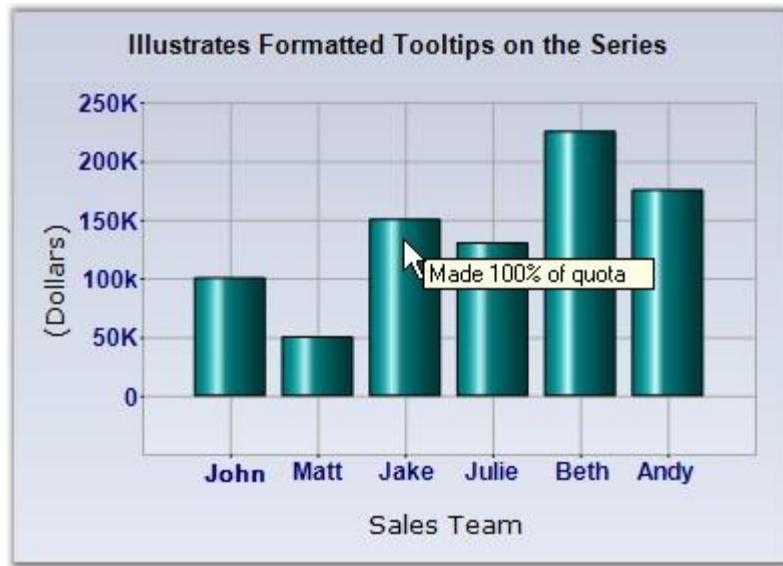


Figure 314: ToolTip Format set for Individual Data Points

### Chart Area Tooltip

Tooltips can also be set for the whole chart area (does not include legends and the space around legends) through the **ChartAreaToolTip**. The data points tooltips will of course override this setting.

### Chart Empty Area Tooltip

The chart also lets you show a tooltip when the mouse hovers over empty areas in the chart (usually around the legend) via the **ChartToolTip** property.

### DataPoint FancyToolTip

Chart Windows includes a "fancy tooltip" feature. As the name implies, this tooltip, which occurs when hovering over a data point looks like a balloon and includes information regarding the series name and the X, Y points. This feature can be turned on by setting the **ChartSeries.FancyToolTip.Visible** property to true.

The FancyToolTip can also be customized with more of the following properties.

FancyToolTip Property	Description
Alignment	Indicates the alignment of the marker to that of the tooltip balloon.

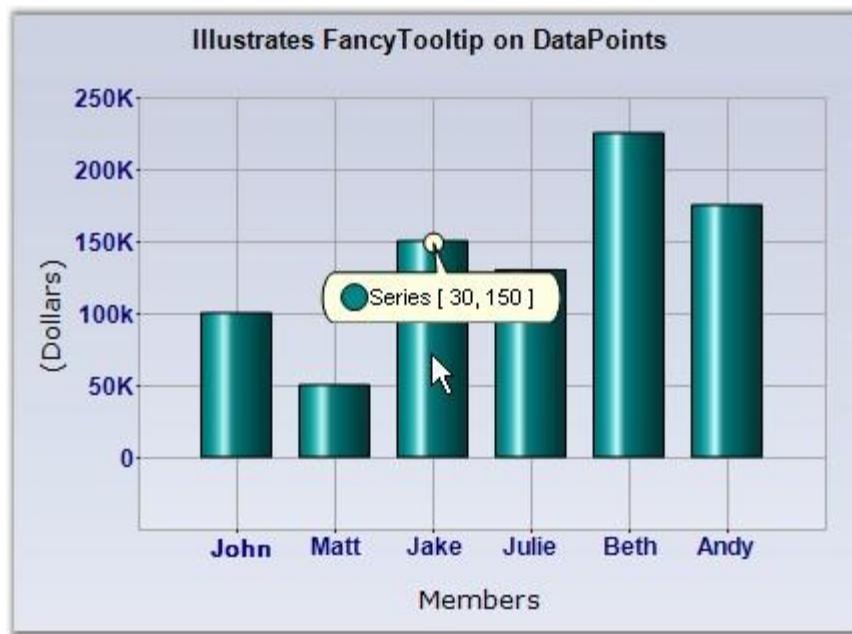
Angle	Specifies the angle at which to render the balloon in the alignment specified.
BackColor	Specifies the back color
Border	Let you customize the border look of the tooltip.
CheckLocation	Specifies whether the tooltip should auto align when shown for data points close to the chart border.
Font	Specifies the font for the tooltip text.
ForeColor	Specifies the color for the tooltip text.
Spacing	The space between the tooltip text and the border.
Style	Specifies the tooltip style. Possible values: Ellipse Rectangle SmoothRectangle - Default value
Symbol	Specifies the symbol shape to use.
SymbolColor	Specifies the inner color of the symbol.
SymbolSize	Specifies the size of the symbol.
ToTarget	Specifies the distance between the balloon and the target.
Visible	Turns on/off fancy tooltips.

**[C#]**

```
series1.FancyToolTip.Visible = true;
series1.FancyToolTip.Alignment = TabAlignment.Top;
```

**[VB .NET]**

```
series1.FancyToolTip.Visible = True
series1.FancyToolTip.Alignment = TabAlignment.Top
```



*Figure 315: Fancy ToolTip set for Data Point*

#### See Also

[How to display tooltip over Histogram Chart columns](#)

## 4.10 Chart Appearance

The following topics under this section discusses about various properties that are used to customize the appearance of the chart:

### 4.10.1 Background Colors

Essential Chart lets you customize the background colors of different portions of the chart.

#### Outside the Chart Area

Use the **BackInterior** property of the chart to customize the background of the chart that is outside the chart area. This is usually where the legend and the chart title get rendered. By default it is set to **White** color.

[C#]

```
this.chartControl1.BackInterior = new  
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.LightBlue);
```

[VB .NET]

```
Me.chartControl1.BackInterior = New  
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.LightBlue)
```

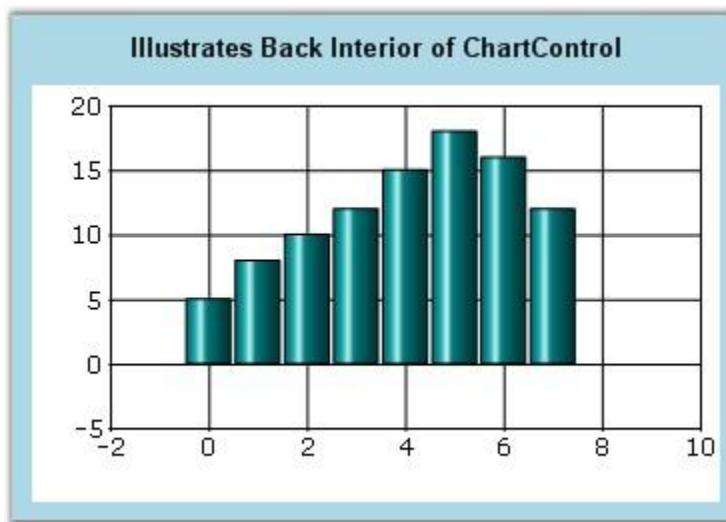


Figure 316: BackInterior = "LightBlue"

### Inside the Plot Area

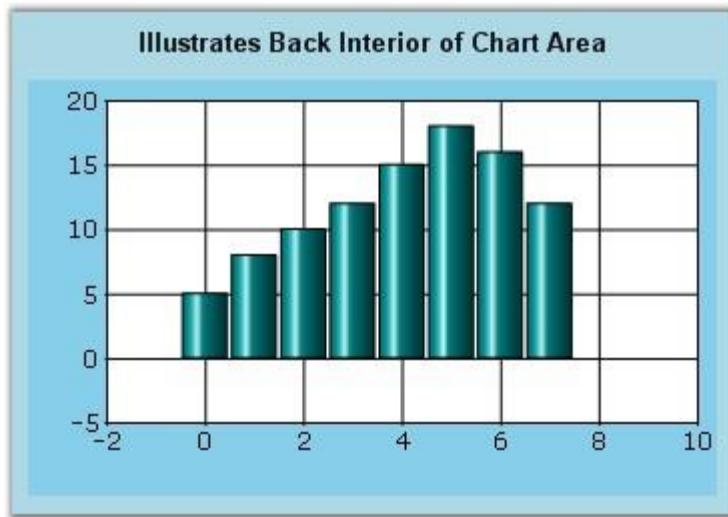
Use the **ChartArea.BackInterior** to customize the background of the rectangular region where the points are plotted.

[C#]

```
this.chartControl1.ChartArea.BackInterior = new  
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.SkyBlue);
```

[VB .NET]

```
Me.chartControl1.ChartArea.BackInterior = New  
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.SkyBlue)
```



*Figure 317: ChartArea.BackInterior = "SkyBlue"*

### **Inside the Chart Area**

Use the **ChartInterior** property of the chart to customize the background of the chart area. By default it is set to **White** color.

#### **[C#]**

```
this.chartControl1.ChartInterior = new  
Syncfusion.Drawing.BrushInfo(Syncfusion.Drawing.GradientStyle.Horizontal  
1, System.Drawing.Color.PaleTurquoise, System.Drawing.Color.LightBlue);
```

#### **[VB .NET]**

```
this.chartControl1.ChartInterior = New  
Syncfusion.Drawing.BrushInfo(Syncfusion.Drawing.GradientStyle.Horizontal  
1, System.Drawing.Color.PaleTurquoise, System.Drawing.Color.LightBlue)
```

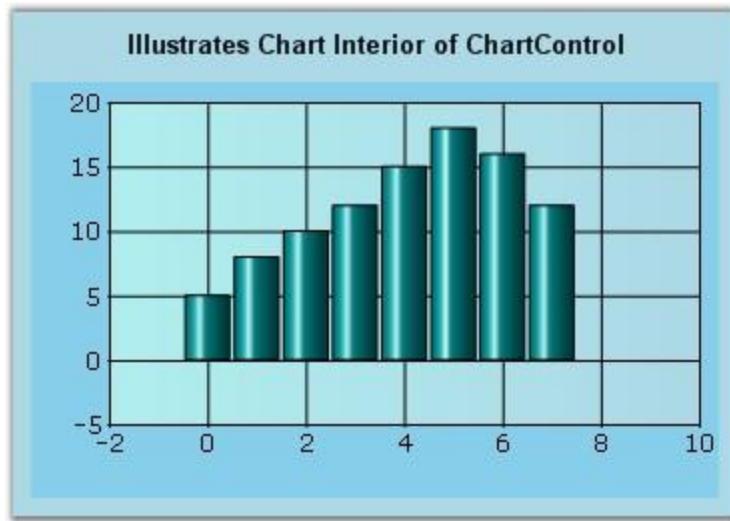


Figure 318: ChartInterior = "LightBlue"

## 4.10.2 Background Image

### Chart Settings

In Windows Forms, use the **BackgroundImage** property to specify a custom image as the background of the chart. The image layout can also be specified using the property below.

Chart control Property	Description
BackgroundImage	Indicates the background image used for the control.
BackgroundImageLayout	Indicates the background image layout used for the component. Possible values are: <b>Tile (default setting)</b> Center Stretch Zoom

[C#]

```
this.chartControl1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("chartControl1.BackgroundImage")));
```

```
this.chartControl1.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Stretch;
```

**[VB .NET]**

```
Me.ChartControl1.BackgroundImageLayout =
 CType((Resources.GetObject("chartControl1.BackgroundImageLayout")),
 System.Drawing.Image)
Me.ChartControl1.BackgroundImageLayout =
 System.Windows.Forms.ImageLayout.Stretch
```

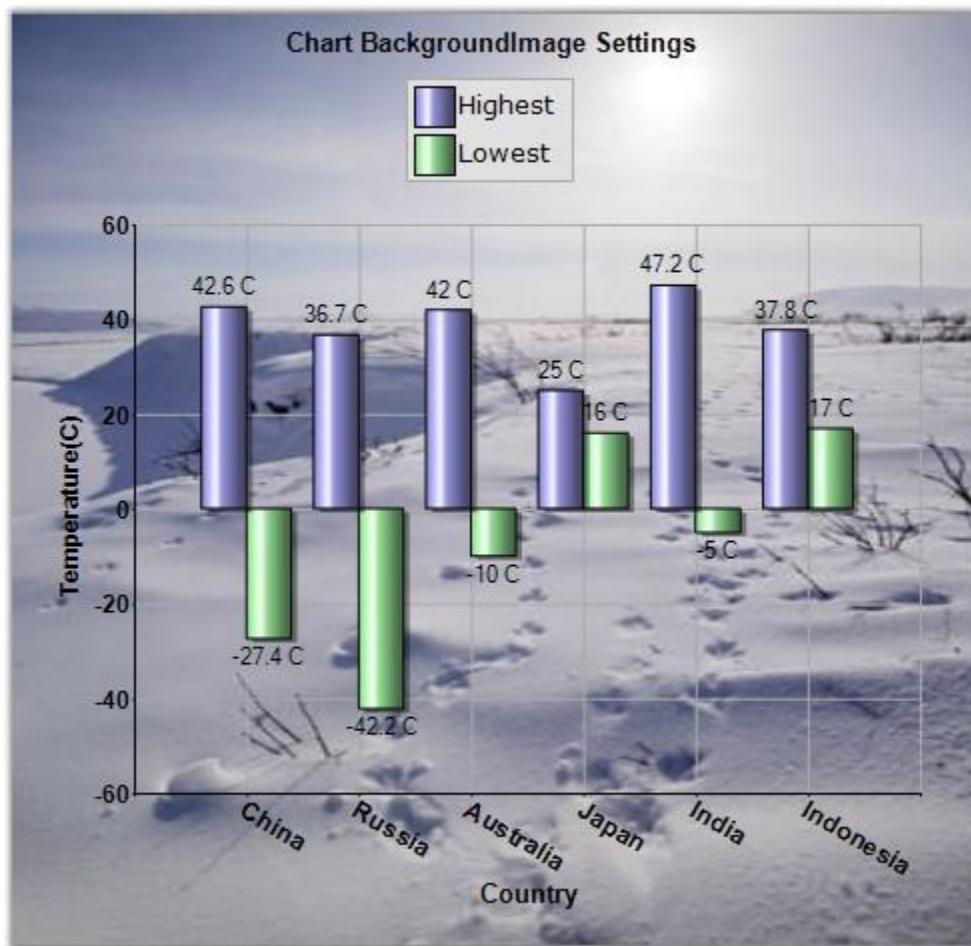


Figure 319: Background Image set for the Chart

### ChartArea Background Image

The chart area can also be rendered with a custom background image and this can be set using the **ChartAreaBackImage** property.

Chart control Property	Description
ChartAreaBackImage	Specifies the image to be used as the background in the chart area.

[C#]

```
this.chartControl1.ChartAreaBackImage = myCustomImage;
```

[VB .NET]

```
Me.ChartControl1.ChartAreaBackImage = myCustomImage
```

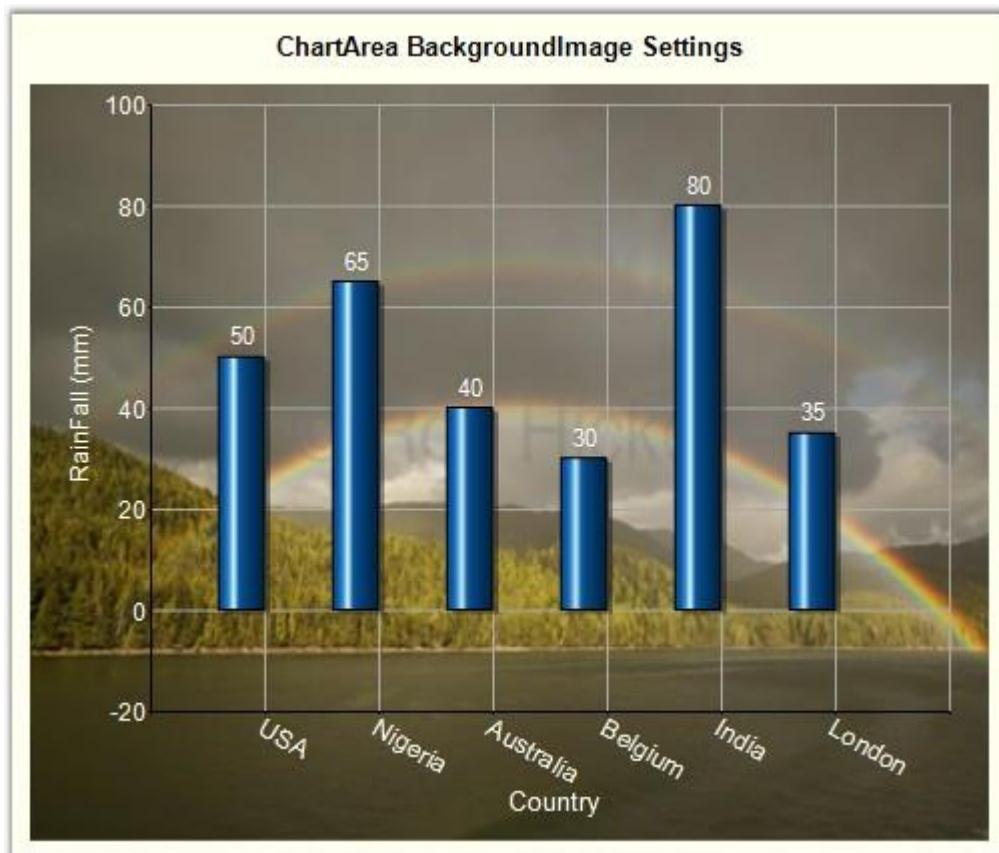


Figure 320: Background Image set for the Chart Area

### Chart Interior Background Image

Chart Interior can be rendered with a custom background image using the **ChartInteriorBackImage** property.

[C#]

```
this.chartControl1.ChartInteriorBackImage = myCustomImage;
```

[VB .NET]

```
Me.ChartControl1.ChartInteriorBackImage = myCustomImage
```

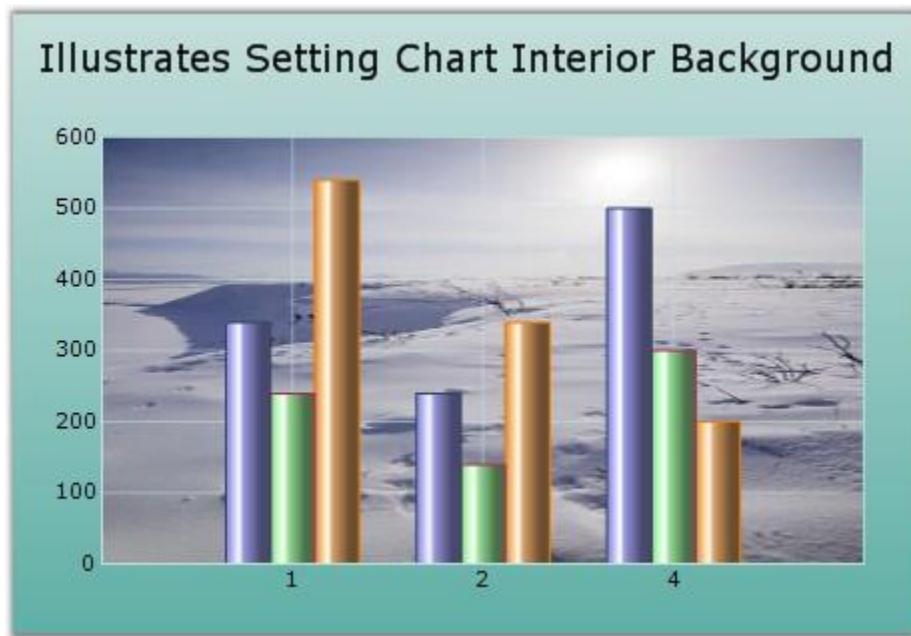


Figure 321: Background Image set for the Chart Interior

### 4.10.3 Border and Margins

#### Chart Area Border

Borders of the chart area can be customized using the below border properties.

ChartArea Property	Description
BorderColor	Indicates the border color of the chart area.

BorderStyle	Indicates the border style.
BorderWidth	Specifies the width of the border.
<b>BorderAppearance</b>	
BaseColor	Gets or sets the color of the base.
FrameThickness	Gets or sets the frame thickness. This property setting will be effective, when SkinStyle is <b>Frame</b> .
Interior	Sets the interior color of the border. This property settings will be effective when SkinStyle is <b>Sunken</b> , <b>Etched</b> and <b>Raised</b> .
SkinStyle	Specifies the border skin style.

**[C#]**

```
this.chartControl1.ChartArea.BorderColor =
System.Drawing.Color.Goldenrod;
this.chartControl1.ChartArea.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
this.chartControl1.ChartArea.BorderWidth = 1;

this.chartControl1.BorderAppearance.BaseColor =
System.Drawing.Color.DarkGray;
//This property setting will be effective, when SkinStyle is 'Frame'.
this.chartControl1.BorderAppearance.FrameThickness = new
Syncfusion.Windows.Forms.Chart.ChartThickness(15F, 30F, 15F, 18F);
//This interior property settings will be effective when SkinStyle is
Sunken, Etched and Raised.
this.chartControl1.BorderAppearance.Interior.ForeColor =
System.Drawing.Color.Maroon;
this.chartControl1.BorderAppearance.SkinStyle =
Syncfusion.Windows.Forms.Chart.ChartBorderSkinStyle.Raised;
```

**[VB .NET]**

```
Me.ChartControl1.ChartArea.BorderColor = System.Drawing.Color.Goldenrod
Me.ChartControl1.ChartArea.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
Me.ChartControl1.ChartArea.BorderWidth = 1

Me.chartControl1.BorderAppearance.BaseColor =
System.Drawing.Color.DarkGray
'This property setting will be effective, when SkinStyle is 'Frame'.
```

```
Me.chartControll1.BorderAppearance.FrameThickness = New  
Syncfusion.Windows.Forms.Chart.ChartThickness(15F, 30F, 15F, 18F)  
'This interior property settings will be effective when SkinStyle is  
Sunken, Etched and Raised.  
Me.chartControll1.BorderAppearance.Interior.ForeColor =  
System.Drawing.Color.Maroon  
Me.chartControll1.BorderAppearance.SkinStyle =  
Syncfusion.Windows.Forms.Chart.ChartBorderSkinStyle.Raised
```

### Chart Area Shadow

The chart area can also be rendered with a shadow. Turn this feature on, by enabling the **ChartAreaShadow** property.

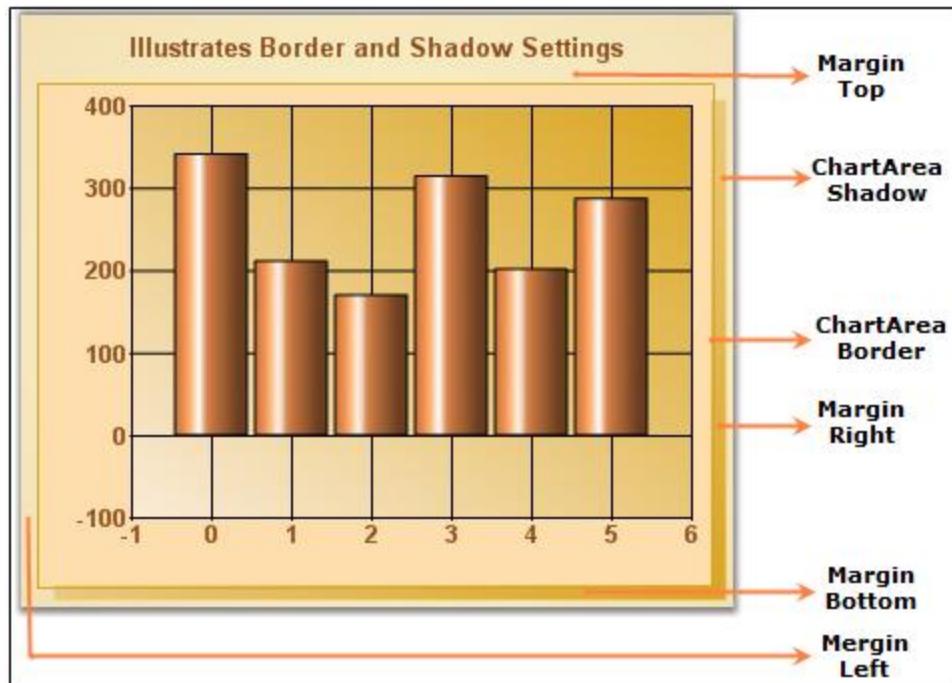
Chart control Property	Description
ChartAreaShadow	Indicates whether chart area has a shadow.
ShadowColor	Specifies the color of the shadow.
ShadowWidth	Specifies the width of the shadow.

#### [C#]

```
this.chartControll1.ChartAreaShadow = true;  
this.chartControll1.ShadowColor = new  
Syncfusion.Drawing.BrushInfo(Syncfusion.Drawing.GradientStyle.ForwardDi  
agonal, System.Drawing.Color.AntiqueWhite,  
System.Drawing.Color.Goldenrod);  
this.chartControll1.ShadowWidth = 7;
```

#### [VB .NET]

```
Me.ChartControll1.ChartAreaShadow = True  
Me.ChartControll1.ShadowColor = New  
Syncfusion.Drawing.BrushInfo(Syncfusion.Drawing.GradientStyle.ForwardDi  
agonal, System.Drawing.Color.AntiqueWhite,  
System.Drawing.Color.Goldenrod)  
Me.chartControll1.ShadowWidth = 7
```

Figure 322: *ChartAreaShadow = "True"*

### Chart Area Margins

Margin for the chart area can be controlled using **ChartAreaMargins** property. It indicates the margin that will be deduced from Chart Area's representation rectangle.

Chart control Property	Description
ChartAreaMargins	Specifies the amount of pixels between the chart area border and the chart plot area. Default is {10, 10, 10, 10}.

#### [C#]

```
this.chartControl1.ChartAreaMargins = new
Syncfusion.Windows.Forms.Chart.ChartMargins(10, 10, 10, 20);
```

#### [VB .NET]

```
Me.ChartControl1.ChartAreaMargins = New
Syncfusion.Windows.Forms.Chart.ChartMargins(10, 10, 10, 20)
```

## ChartPlot Area Margins

The margins for ChartPlotArea is specified in **ChartPlotAreaMargins** property.

ChartControl Property	Description
ChartPlotAreaMargins	Indicates the margin of the axis labels. This margin is supported for left, Top, Right and Bottom side of the chart. This property works only if <b>EdgeLabelsDrawingMode</b> property is set to <b>Shift</b> .  Default is {10, 10, 10, 10}.
AdjustPlotAreaMargins	Gets / sets the mode of drawing the edge labels. Default is <b>AutoSet</b> .
EdgeLabelsDrawingMode	Gets or sets the edge labels drawing mode.

### [C#]

```
this.chartControll1.PrimaryYAxis.EdgeLabelsDrawingMode =
Syncfusion.Windows.Forms.Chart.ChartAxisEdgeLabelsDrawingMode.Shift;
this.chartControll1.ChartArea.AdjustPlotAreaMargins =
Syncfusion.Windows.Forms.Chart.ChartSetMode.UserSet;
this.chartControll1.ChartArea.ChartPlotAreaMargins.Left = 200;
```

### [VB .NET]

```
Me.chartControll1.PrimaryYAxis.EdgeLabelsDrawingMode =
Syncfusion.Windows.Forms.Chart.ChartAxisEdgeLabelsDrawingMode.Shift
Me.chartControll1.ChartArea.AdjustPlotAreaMargins =
Syncfusion.Windows.Forms.Chart.ChartSetMode.UserSet
Me.chartControll1.ChartArea.ChartPlotAreaMargins.Left = 200
```

## Spacing between elements

The spacing between elements in the chart is specified using the **ElementsSpacing** property. For example, the space between the chart right border and legend right border if **LegendPosition** is set to **Right**.

Chart control Property	Description
ElementsSpacing	Specifies the spacing between the elements in the chart. Default is 20.

## 4.10.4 Foreground Settings

### Chart Title

The ChartControl provides properties to customize and align the text within the control. Below are the text properties.

Using the **ChartControl.Text** property, users can provide the title that appears at the top of the chart. **TextPosition** and **TextAlignment** further lets you control the relative positioning of this title.

Here are some properties that affect the title text in the chart.

Chart control Property	Description
Text	Specifies the title for the chart.
TextPosition	Specifies the position of the chart. Possible values are, <ul style="list-style-type: none"> <li>• Top (<b>default setting</b>)</li> <li>• Bottom</li> <li>• Left</li> <li>• Right</li> </ul>
TextAlignment	Specifies the alignment of the title with respect to the chart borders. Possible values: <ul style="list-style-type: none"> <li>• Near</li> <li>• Center (<b>default setting</b>)</li> <li>• Far</li> </ul>
Font	Indicates the font style of the title.
ForeColor	Indicates the foreground color of the title.

[C#]

```
this.chartControl1.Text = "Illustrates Foreground Settings";
```

```
this.chartControl1.Font = new System.Drawing.Font("Arial", 11.25F,  
System.Drawing.FontStyle.Bold);  
this.chartControl1.ForeColor = System.Drawing.Color.Bisque;  
this.chartControl1.TextPosition = ChartTextPosition.Top;
```

**[VB .NET]**

```
Me.ChartControl1.Text = "Illustrates Foreground Settings"  
Me.chartControl1.Font = New System.Drawing.Font("Arial", 11.25F,  
System.Drawing.FontStyle.Bold)  
Me.chartControl1.ForeColor = System.Drawing.Color.Bisque  
Me.chartControl1.TextPosition = ChartTextPosition.Top
```

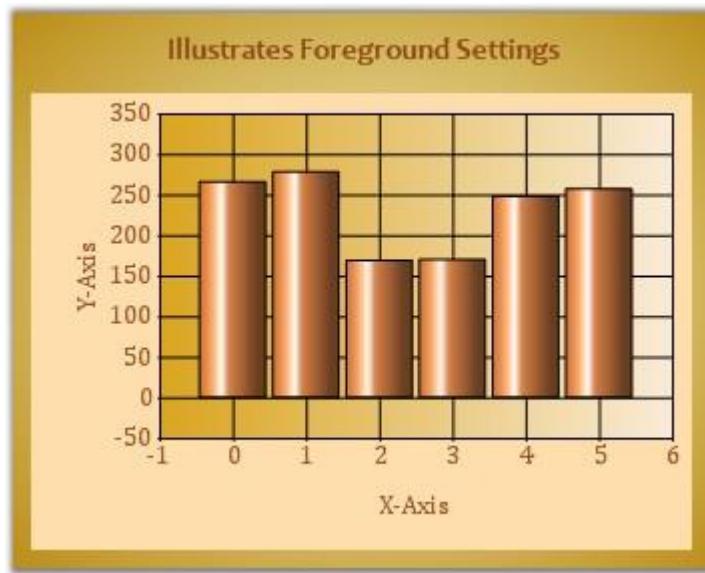


Figure 323: Illustrates changes affecting the Title Text

### General Text Related settings

The following text related properties affect all the text rendered in the chart.

Chart control Property	Description
TextRenderingHint	Specifies the way the text is drawn. Possible values: <ul style="list-style-type: none"><li>• <b>AntiAlias</b> - each character is drawn using its anti-aliased glyph bitmap without hinting.</li><li>• <b>AntiAliasGridFit</b> - each character is drawn using its anti-aliased</li></ul>

	<p>glyph bitmap with hinting.</p> <ul style="list-style-type: none"> <li>• <b>ClearTypeGridFit</b> - each character is drawn using its glyph clear type bitmap with hinting.</li> <li>• <b>SingleBitPerPixel</b> - each character is drawn using its glyph bitmap.</li> <li>• <b>SingleBitPerPixelGridFit</b> - each character is drawn using its glyph bitmap.</li> <li>• <b>SystemDefault</b> - each character is drawn using its glyph bitmap with the system default rendering hint. The text will be drawn using whatever the font-smoothing settings the user had selected for the system. (<b>default setting</b>)</li> </ul>
SmoothingMode	Specifies how chart elements should be rendered. Possible values: <ul style="list-style-type: none"> <li>• AntiAlias</li> <li>• HighQuality</li> <li>• HighSpeed</li> <li>• Invalid</li> <li>• None</li> <li>• Default(<b>default setting</b>)</li> </ul>

### See Also

[Axis Label Text Formatting, Appearance and Positioning](#), (for info on changing axis label text settings)

[Customizing Label Text, Intersecting Labels, Grouping Labels](#), (for info on changing axis label text settings)

[Series Customization/Font](#), (for info on changing series text settings)

[Chart Legend](#) (for info on changing legend text settings)

## 4.10.5 Multiple Chart Titles

### Default Title

Essential Chart's **Title** property lets you edit the default title for a chart as follows. We can set font style for the title using **Title.Font** property. The default value is Verdana, 14, Regular.

[C#]

```
//Default title  
chartControl1.Title.Text = "Essential Chart";  
this.chartControl1.Title.Font = new System.Drawing.Font("Candara", 9F,  
System.Drawing.FontStyle.Bold);
```

[VB.NET]

```
'Default title  
chartControl1.Title.Text = "Essential Chart"  
chartControl1.Title.Font = New System.Drawing.Font("Candara", 9F,  
System.Drawing.FontStyle.Bold)
```

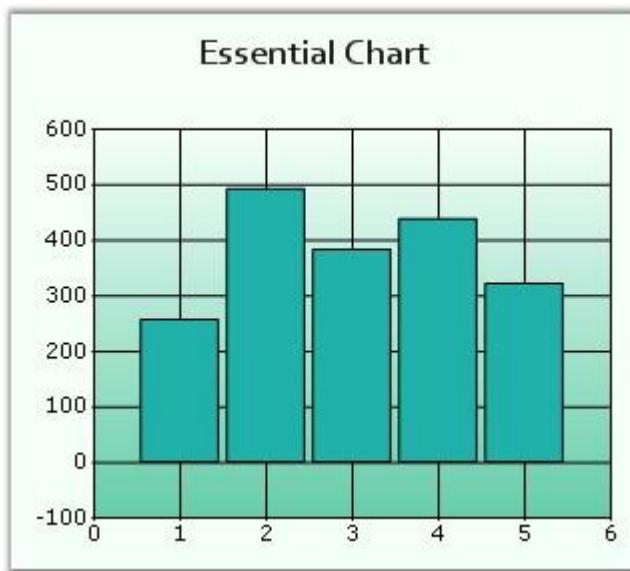


Figure 324: Chart Title Set

The above default chart title is simply the first in the list of titles that can be specified for the Chart.

### Multiple Titles

- Multiple custom Chart Titles can be added to **Chart.Titles** Collection.
- Supports numerous docking styles (Floating, Left, Right, Bottom or Top) for each title.
- Each of the custom Titles can be aligned to any position as required.

### Titles Positioning

Below listed properties will help you to modify the positioning of the Chart Title.

ChartTitle Property	Description
Position	<p>Specifies the position relative to the chart at which to render the chart title panel.</p> <ul style="list-style-type: none"><li>• <b>Top</b> - above the chart(<b>Default setting</b>)</li><li>• <b>Left</b> - left of the chart</li><li>• <b>Right</b> - right of the chart</li><li>• <b>Bottom</b> - below the chart</li><li>• <b>Floating</b> - will not be docked to any specific location. Can be docked manually by dragging the title panel.</li></ul>
Alignment	<p>When docked to a side, this property specifies how the title panel should be aligned with respect to the chart boundaries.</p> <ul style="list-style-type: none"><li>• <b>Center</b> - will be aligned to center. <b>Default setting</b>.</li><li>• <b>Far</b> - will be aligned Far.</li><li>• <b>Near</b> - will be aligned Near.</li></ul>
Behavior	<p>Specifies the docking behavior of the title.</p> <ul style="list-style-type: none"><li>• <b>Docking</b> - It is dockable on all four sides.</li><li>• <b>Movable</b> - It is movable.</li><li>• <b>All</b> - It is movable and dockable.</li><li>• <b>None</b> - It is neither movable nor dockable.</li></ul>

### **Title Look and Feel**

There are several appearance options that can be applied on the ChartTitle instance as illustrated in this **ChartTitle Collection Editor**.

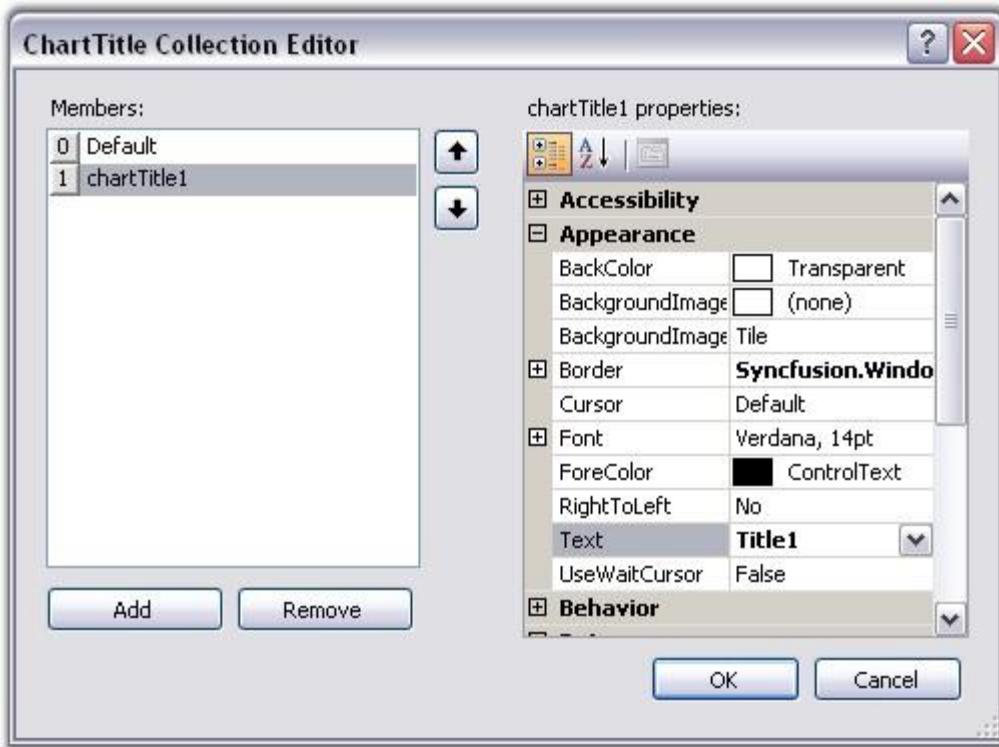


Figure 325: ChartTitle Collection Editor

In code, you can add more titles to this list as follows.

**[C#]**

```
//Default title (the first entry in the Titles list)
chartControl1.Title.Text = "Essential Chart";

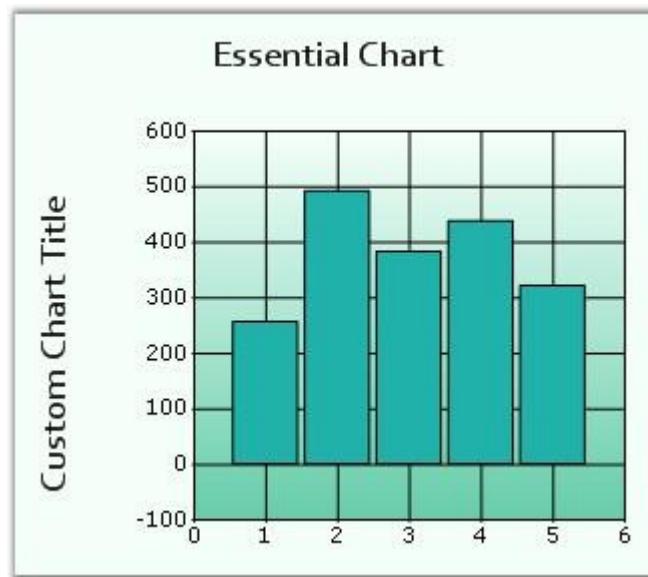
// Add the title to the Chart control's Titles collection.
ChartTitle title = new Syncfusion.Windows.Forms.Chart.ChartTitle();
title.Text = "Custom Chart Title";
this.chartControl1.Titles.Add(title);
```

**[VB.NET]**

```
'Default title (the first entry in the Titles list)
chartControl1.Title.Text = "Essential Chart"

' Add the title to the Chart control's Titles collection.
Dim title As New Syncfusion.Windows.Forms.Chart.ChartTitle
title.Text = "Custom Chart Title"
```

```
Me.ChartControll.Titles.Add(title)
```



*Figure 326: Chart with Multiple Chart Titles*

#### **Multiline Chart Title**

You can now wrap the Chart titles and display them as multiline text. Set multiline title text in **ChartTitle.Text** property through designer as follows. Press ENTER key to begin a new line. Press CTRL+ENTER to set the text entered.

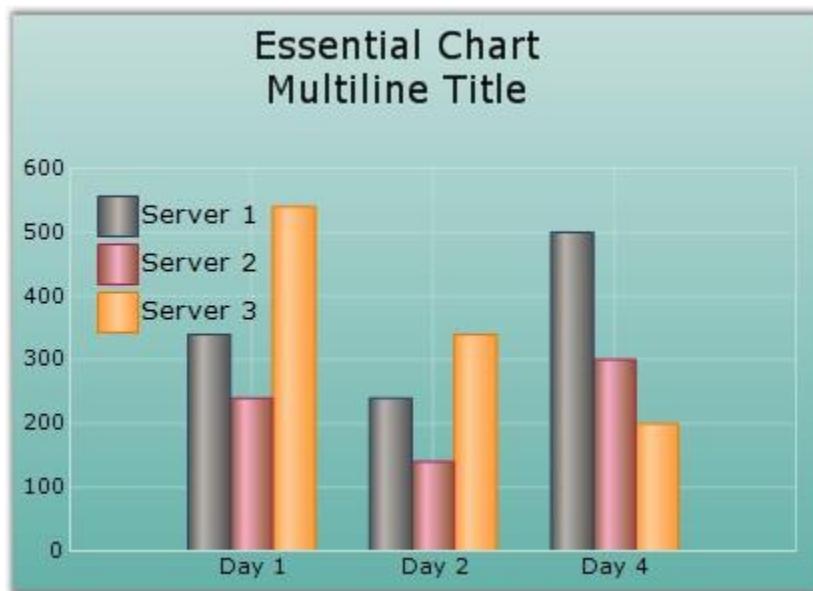


Figure 327: Multiline Title for Essential Chart

## 4.10.6 Custom Drawing

Essential Chart lets you render any data on the chart area. If the built-in features and functionality are not sufficient you can simply draw whatever you want on the chart surface.

You can do so by listening to the **ChartAreaPaint** event. This event is raised both when a chart is painted as well as when the chart is exported to other image formats, SVG, etc. Remember to do your custom drawing in this event instead of in the **Paint** event (which will not be called during chart export).

```
[C#]

private void chartControl1_ChartAreaPaint(object sender, PaintEventArgs e)
{
    // Get the right end of the X axis
    Point ptX = this.chartControl1.ChartArea.GetPointByValue(new
    ChartPoint(this.chartControl1.PrimaryXAxis.Range.Max,
    this.chartControl1.PrimaryYAxis.Range.Min));

    PointF ptx1 = new PointF(ptX.X - 7, ptX.Y - 4);
    PointF ptx2 = new PointF(ptX.X, ptX.Y);
    PointF ptx3 = new PointF(ptX.X + 7, ptX.Y + 4);

    // Draws an arrow at the end of the X axis
    e.Graphics.FillPolygon(Brushes.Black, new PointF[] { ptx1, ptx2,
    ptx3 });

    // Get the top end of the Y axis
    Point ptY = this.chartControl1.ChartArea.GetPointByValue(new
    ChartPoint(this.chartControl1.PrimaryXAxis.Range.Min,
    this.chartControl1.PrimaryYAxis.Range.Max));

    PointF pty1 = new PointF(ptY.X - 4, ptY.Y + 7);
    PointF pty2 = new PointF(ptY.X, ptY.Y);
    PointF pty3 = new PointF(ptY.X + 4, ptY.Y + 7);

    // Draws an arrow at the top of the Y Axis.
    e.Graphics.FillPolygon(Brushes.Black, new PointF[] { pty1, pty2,
    pty3 });

    // Draws a line through the center of the chart.
    e.Graphics.DrawLine(Pens.Gray, ptY.X, ptX.Y, ptx.X, pty.Y);
}
```

```
}
```

**[VB.NET]**

```
Private Sub chartControl1_ChartAreaPaint(ByVal sender As Object, ByVal e As PaintEventArgs)
    ' Get the right end of the X axis
    Dim ptX As Point = Me.chartControl1.ChartArea.GetPointByValue(New ChartPoint(Me.chartControl1.PrimaryXAxis.Range.Max,
Me.chartControl1.PrimaryYAxis.Range.Min))

    Dim ptX1 As New PointF(ptX.X - 7, ptX.Y - 4)
    Dim ptX2 As New PointF(ptX.X, ptX.Y)
    Dim ptX3 As New PointF(ptX.X - 7, ptX.Y + 4)

    ' Draws an arrow at the end of the X axis
    e.Graphics.FillPolygon(Brushes.Black, New PointF() {ptX1, ptX2,
ptX3})

    ' Get the top end of the Y axis
    Dim ptY As Point = Me.chartControl1.ChartArea.GetPointByValue(New ChartPoint(Me.chartControl1.PrimaryXAxis.Range.Min,
Me.chartControl1.PrimaryYAxis.Range.Max))

    Dim ptY1 As New PointF(ptY.X - 4, ptY.Y + 7)
    Dim ptY2 As New PointF(ptY.X, ptY.Y)
    Dim ptY3 As New PointF(ptY.X + 4, ptY.Y + 7)

    ' Draws an arrow at the top of the Y Axis.
    e.Graphics.FillPolygon(Brushes.Black, New PointF() {ptY1, ptY2,
ptY3})

    ' Draws a line through the center of the chart.
    e.Graphics.DrawLine(Pens.Gray, ptY.X, ptX.Y, ptX.X, ptY.Y)
End Sub
```

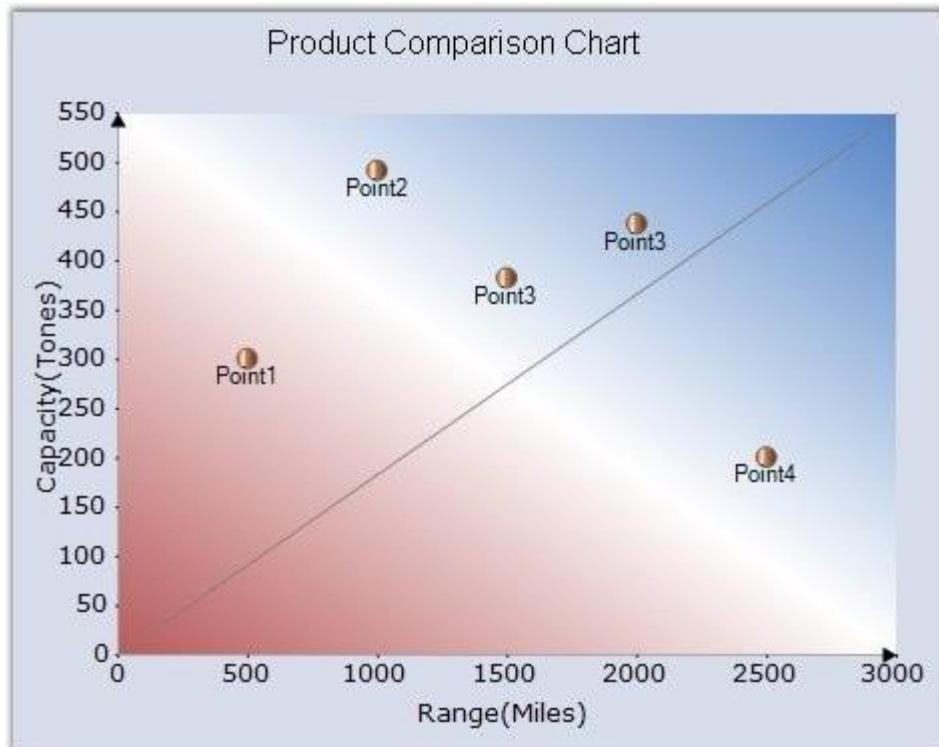


Figure 328: Chart with Custom Drawing - Arrows at the end of the Axes and a Diagonal Line

#### See Also

[Chart Area Bounds](#)

#### 4.10.7 Watermark Support

Essential Chart supports watermark feature using which we can show text, image, or both as watermark inside the chart area.

Below are the WaterMark properties with descriptions.

ChartAxis Property	Description
Text	Sets the watermark text.
Image	Used to display image as the watermark.
Opacity	Sets the opacity of the watermark.

HorizontalAlign	Sets watermark horizontally in the chart area.
VerticalAlign	Sets watermark vertically in the chart area.
Zorder	Used to specify whether watermark should be shown on top of the chart.

**[C#]**

```
this.chartControl1.ChartArea.WaterMark.Text="Syncfusion Chart";
this.chartControl1.ChartArea.Watermark.Image =
System.Drawing.Image.FromFile("Logo.bmp");
this.chartControl1.ChartArea.Watermark.Opacity=60;
this.chartControl1.ChartArea.Watermark.HorizontalAlignment=ChartAlignment.Near
this.chartControl1.ChartArea.Watermark.VerticalAlignment=ChartAlignment.Near;
this.chartControl1.ChartArea.Watermark.ZOrder=ChartWaterMarkOrder.Behind;
```

**[VB .NET]**

```
Me.chartControl1.ChartArea.WaterMark.Text="Syncfusion Chart"
Me.chartControl1.ChartArea.Watermark.Image =
System.Drawing.Image.FromFile("Logo.bmp")
Me.chartControl1.ChartArea.Watermark.Opacity=60
Me.chartControl1.ChartArea.Watermark.HorizontalAlignment=ChartAlignment.Near
Me.chartControl1.ChartArea.Watermark.VerticalAlignment=ChartAlignment.Near
Me.chartControl1.ChartArea.Watermark.ZOrder=ChartWaterMarkOrder.Behind;
```



Figure 329: "Image" displayed as Watermark; Opacity = "60"; HorizontalAlignment = "Near"; VerticalAlignment = "Near"; ZOrder = "Behind"

#### 4.10.8 Interlaced Grid Background

Chart supports interlaced grid which draws alternative grid background in x-axis and y-axis. The color is also customizable.

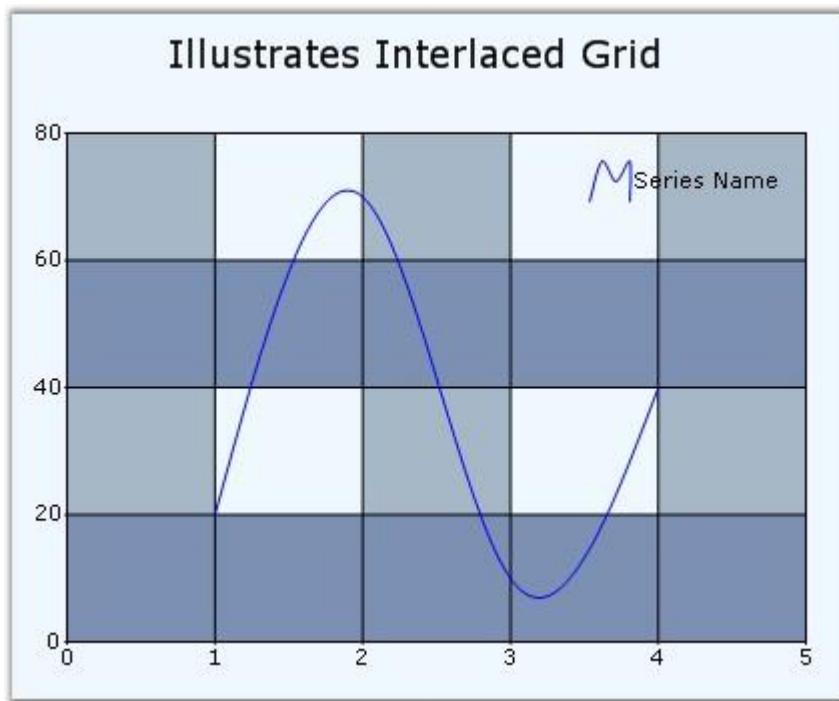
##### [C#]

```
this.chartControl1.PrimaryXAxis.InterlacedGrid = true;
this.chartControl1.PrimaryXAxis.InterlacedGridInterior = new
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.FromArgb(166, 184,
200));
this.chartControl1.PrimaryYAxis.InterlacedGrid = True;
this.chartControl1.PrimaryYAxis.InterlacedGridInterior = new
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.FromArgb(124, 144,
179));
```

##### [VB .NET]

```
Me.chartControl1.PrimaryXAxis.InterlacedGrid = True
Me.chartControl1.PrimaryXAxis.InterlacedGridInterior = new
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.FromArgb(166, 184,
```

```
200)
Me.chartControll.PrimaryYAxis.InterlacedGrid = True
Me.chartControll.PrimaryYAxis.InterlacedGridInterior = new
Syncfusion.Drawing.BrushInfo(System.Drawing.Color.FromArgb(124, 144,
179))
```



*Figure 330: Interlaced Grid*

The preceding image illustrates interlaced grid background for the chart.

A sample which illustrates the Interlaced Grid for the Chart is available in the below sample installation location.

[<Sample location>\Syncfusion\EssentialStudio\Version  
Number\Windows\Chart.Windows\Samples\2.0\Chart Appearance\Interlaced Grid](<Sample location>\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Chart Appearance\Interlaced Grid)

#### 4.10.9 Minor Grid Lines

Chart comes with minor lines support which will draw lines along the intervals provided. The appearance of these line is also customizable similar to the major grid lines.

```
[C#]
```

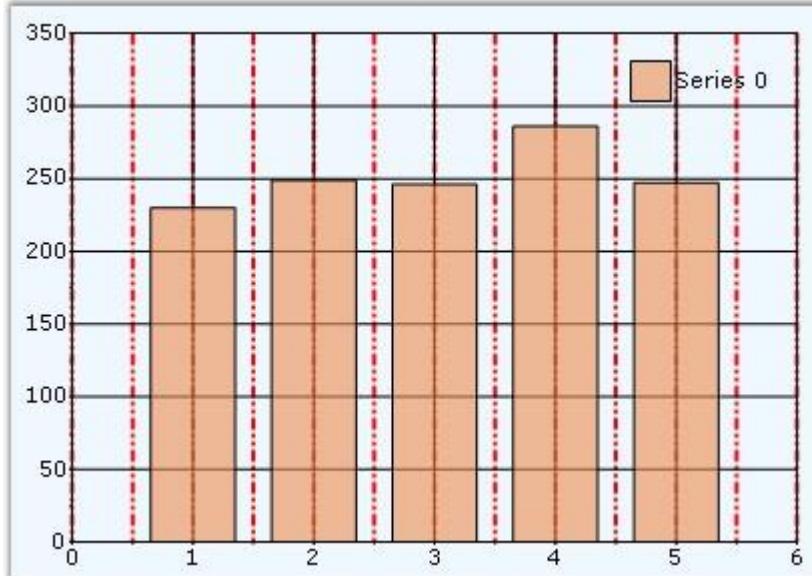
```
this.chartControl1.PrimaryXAxis.DrawMinorGrid = true;
this.chartControl1.PrimaryXAxis.MinorGridLineType.DashStyle =
DashStyle.DashDotDot;
this.chartControl1.PrimaryXAxis.MinorGridLineType.Width = 2;
this.chartControl1.PrimaryXAxis.MinorGridLineType.ForeColor =
Color.Red;
chartControl1.PrimaryXAxis.SmallTicksPerInterval = 1;
```

**[VB .NET]**

```
Me.chartControl1.PrimaryXAxis.DrawMinorGrid = True
Me.chartControl1.PrimaryXAxis.MinorGridLineType.DashStyle =
DashStyle.DashDotDot
Me.chartControl1.PrimaryXAxis.MinorGridLineType.Width = 2
Me.chartControl1.PrimaryXAxis.MinorGridLineType.ForeColor = Color.Red
chartControl1.PrimaryXAxis.SmallTicksPerInterval = 1
```



**Note:** In the above code we have specified value for `SmallTicksPerInterval` property. No of minor grids lines depends on the value of this property of Chart Axis. Default value is 0; So, MinorGridLines will not appear in the chart by default. To see the minor grid lines in the chart, set `SmallTicksPerInterval` property to 1 or greater than 1.



*Figure 331*

The preceding image illustrates custom minor grid lines on x-axis.

#### 4.10.10 Chart Skins

Chart Control has some pre-defined skins, which can be controlled through a single property setting. Essential Chart control allows the user to customize its appearance by applying pre-defined Interiors.

Some of the available skins are:

- Office2007Black
- Office2007Blue
- Office2007Silver
- Almond
- Blend
- Blueberry
- Marble
- Midnight
- Monochrome
- Olive
- Sandune
- Turquoise
- Vista
- VS2010

[C#]

```
this.chartControl1.Skins = Skins.Office2007Blue;
```

[VB]

```
Me.chartControl1.Skins = Skins.Office2007Blue
```

The following output is displayed when the Skins value is set to Office2007 Black.

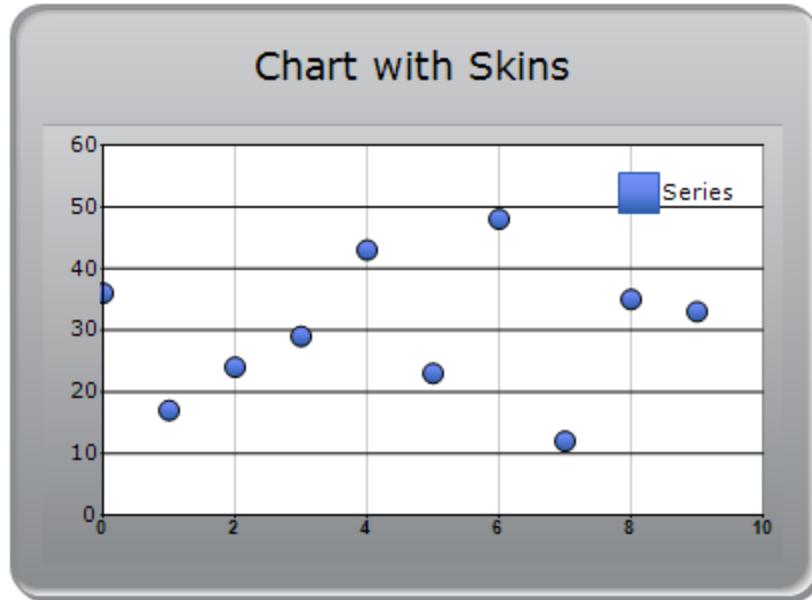


Figure 332: Office2007 Black

The following output is displayed when the Skins value is set to Office2007 Blue.

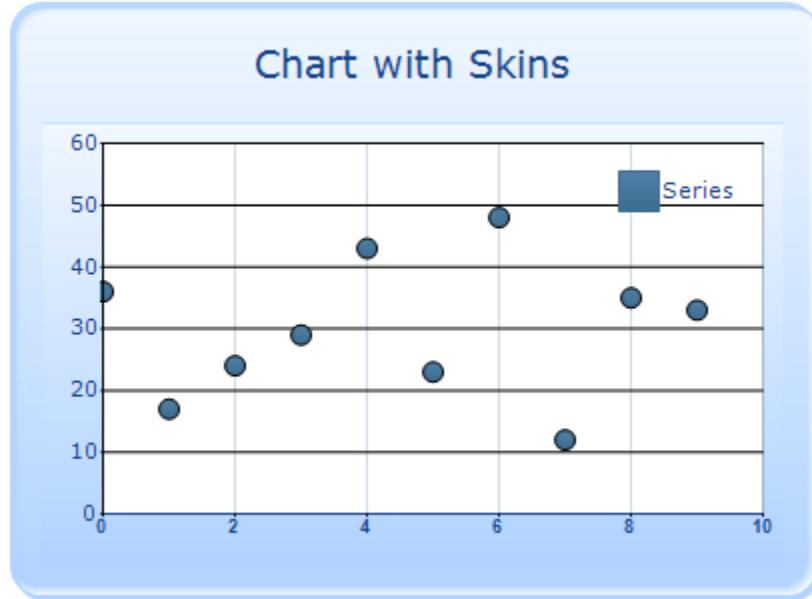


Figure 333: Office2007 Blue

The following output is displayed when the Skins value is set to Office2007 Silver.

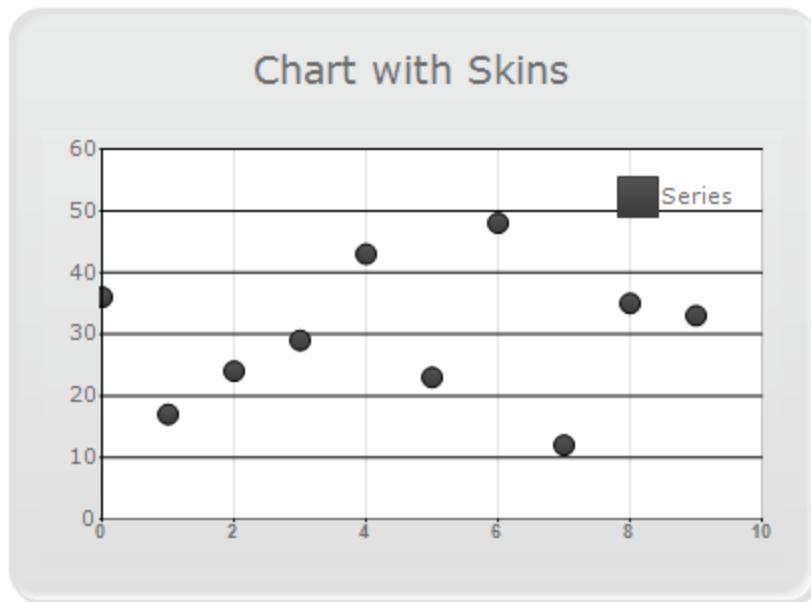


Figure 334: Office2007 Silver

The following output is displayed when the Skins value is set to Almond.

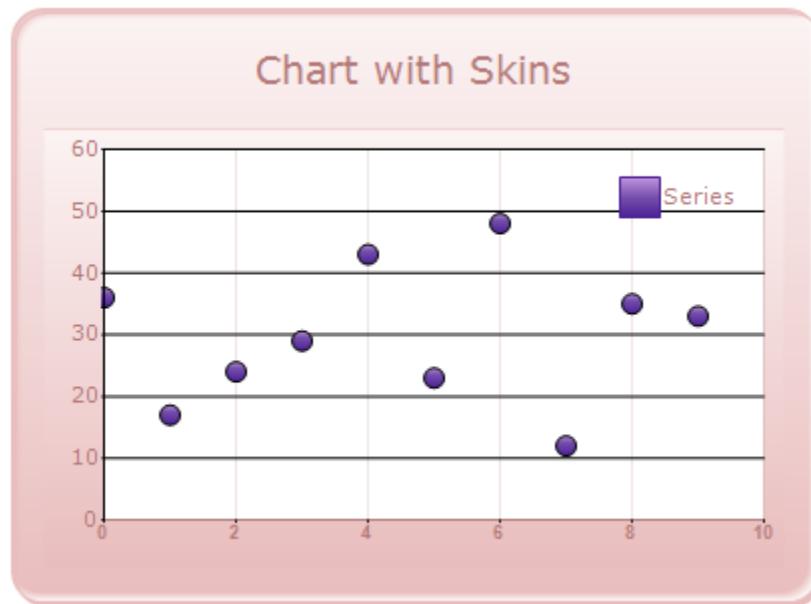
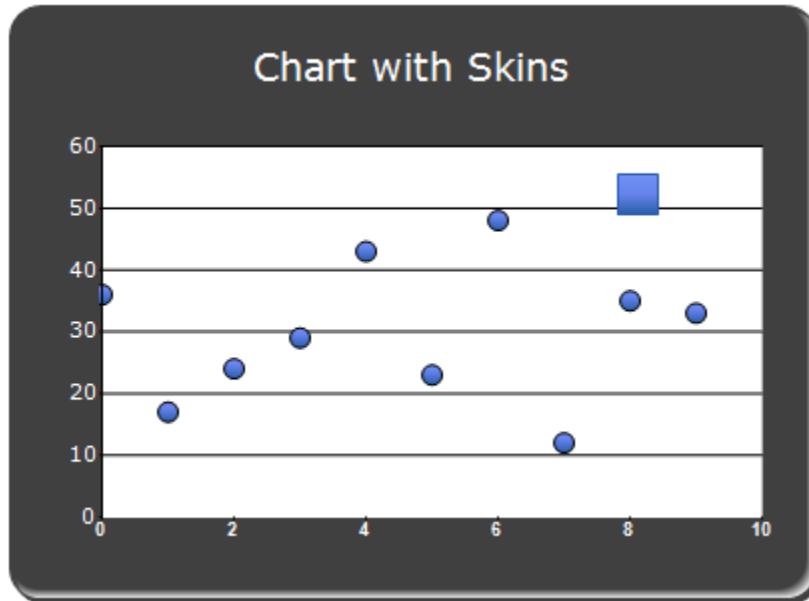


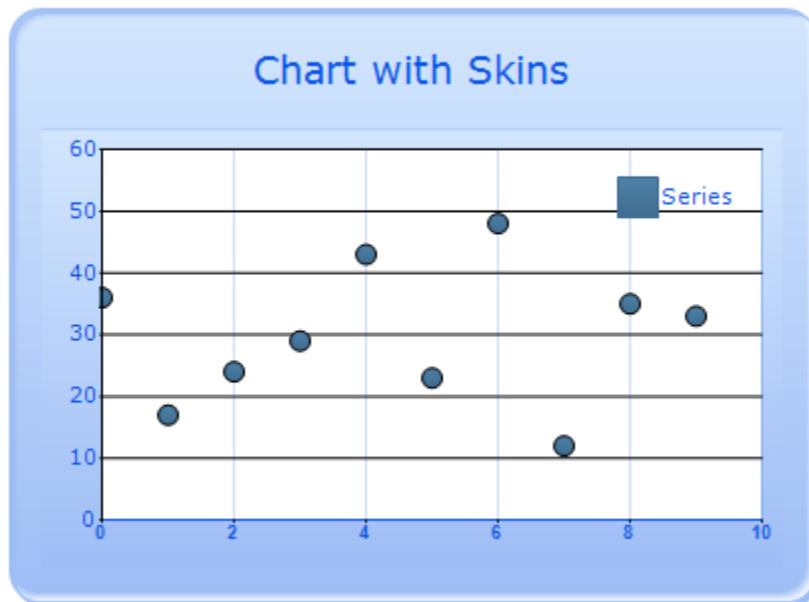
Figure 335: Almond

The following output is displayed when the Skins value is set to Blend.



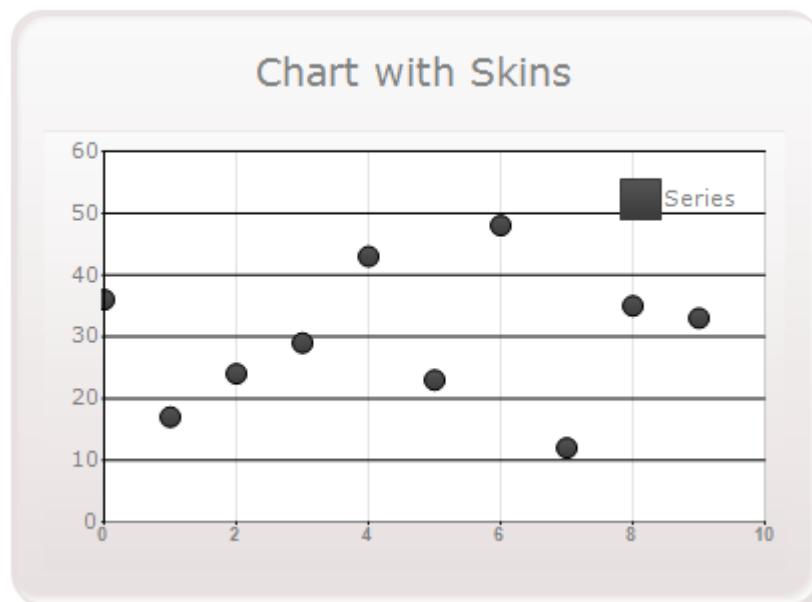
*Figure 336: Blend*

The following output is displayed when the Skins value is set to Blueberry.



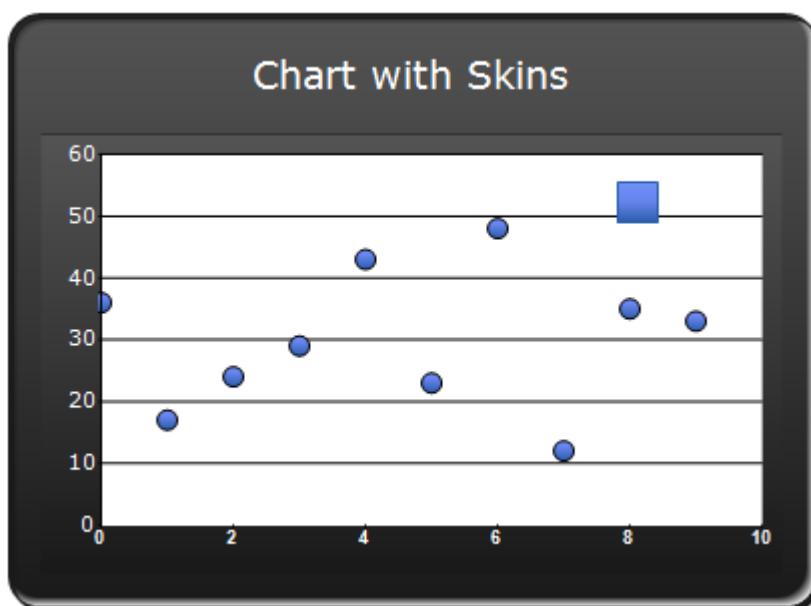
*Figure 337: Blueberry*

The following output is displayed when the Skins value is set to Marble.



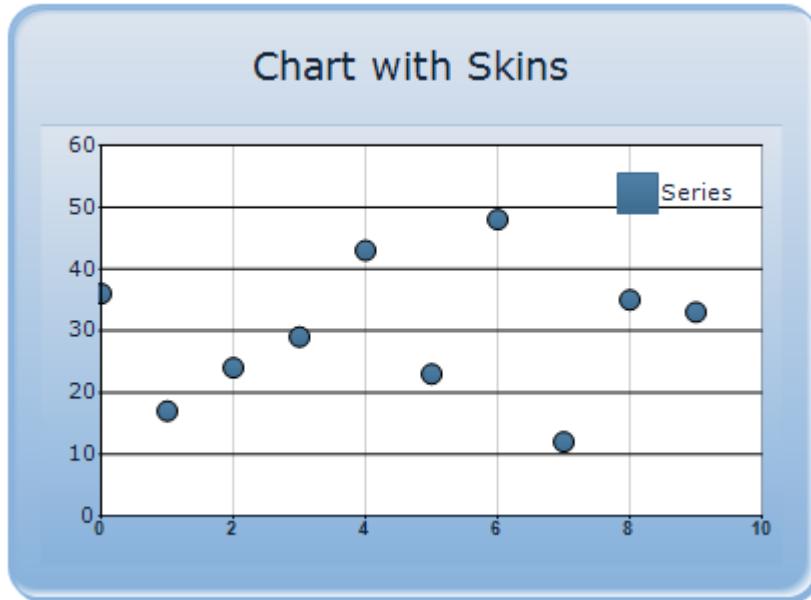
*Figure 338: Marble*

The following output is displayed when the Skins value is set to Midnight.



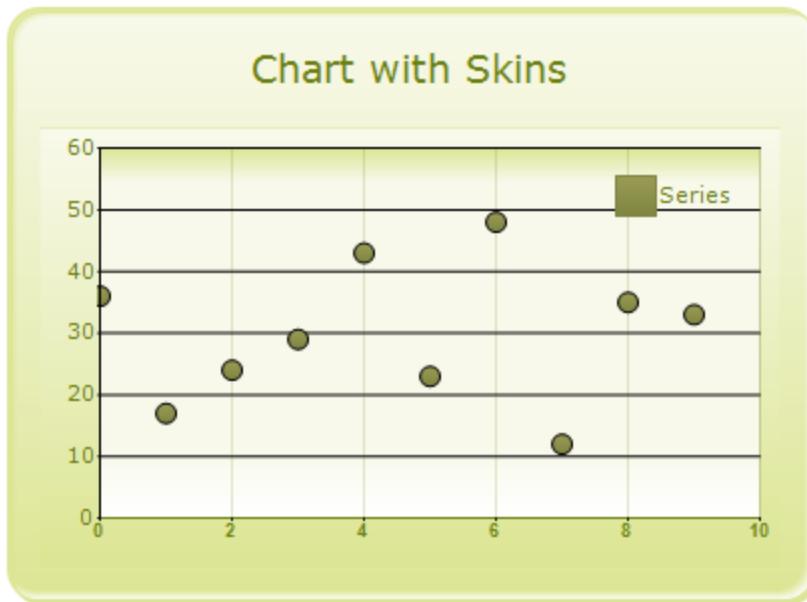
*Figure 339: Midnight*

The following output is displayed when the Skins value is set to Monochrome.



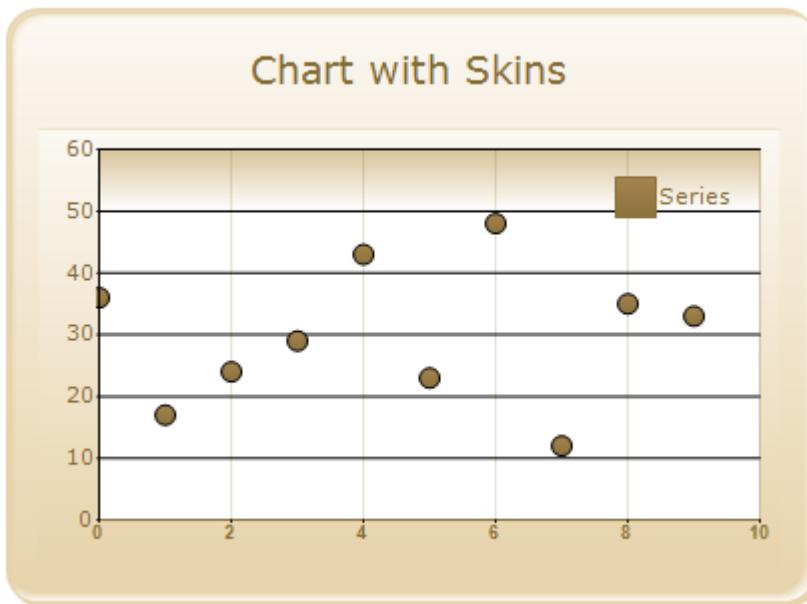
*Figure 340: Monochrome*

The following output is displayed when the Skins value is set to Olive.



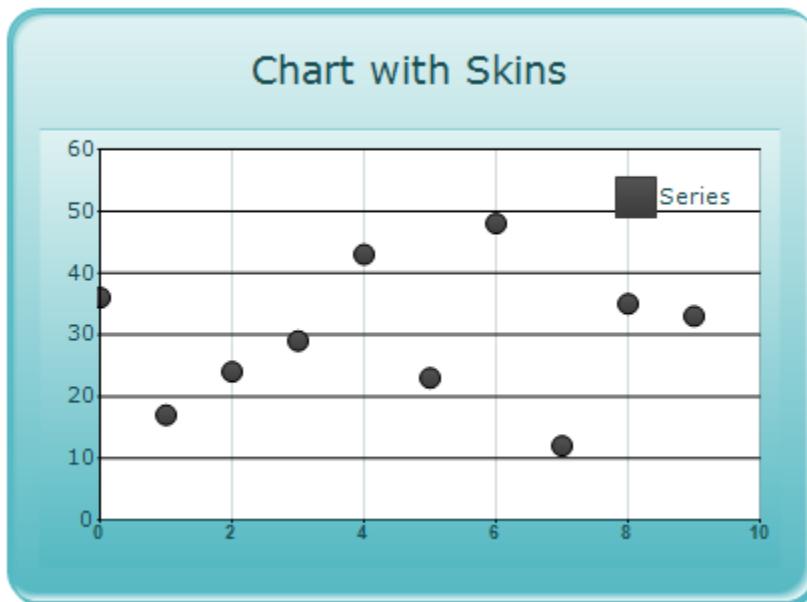
*Figure 341: Olive*

The following output is displayed when the Skins value is set to Sandune.



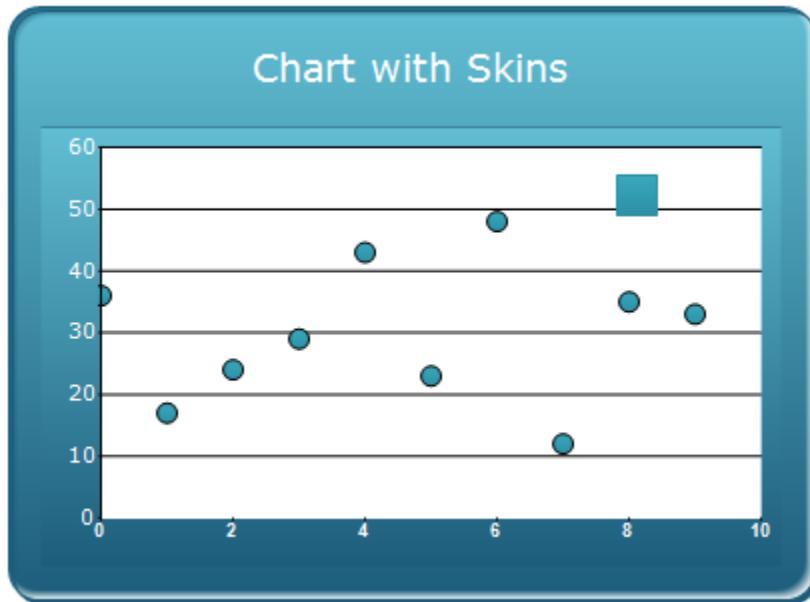
*Figure 342: Sandune*

The following output is displayed when the Skins value is set to Turquoise.



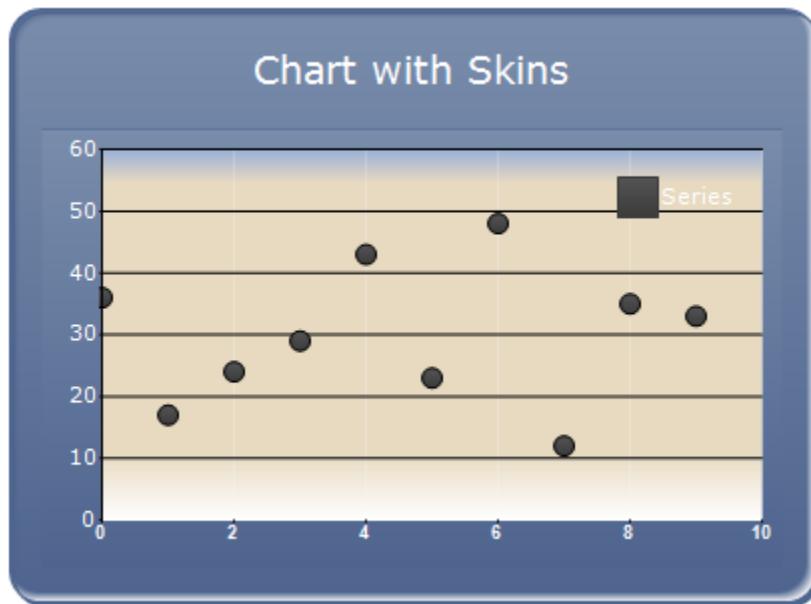
*Figure 343: Turquoise*

The following output is displayed when the Skins value is set to Vista.



*Figure 344: Vista*

The following output is displayed when the Skins value is set to VS2010.



*Figure 345: VS2010*

[C#]

```
ChartSeries ser1 = new ChartSeries("Series 1");
ser1.Type = ChartSeriesType.StackingColumn;
// specifying group name .
ser1.StackingGroup = "FirstGroup";
ChartSeries ser2 = new ChartSeries("Series 2");
ser2.Type = ChartSeriesType.StackingColumn;
// specifying group name .
ser2.StackingGroup = "SecondGroup";
ChartSeries ser3 = new ChartSeries("Series 3");
```

```
ser3.Type = ChartSeriesType.StackingColumn;
// specifying group name .
ser3.StackingGroup = "FirstGroup";
```

[VB]

```
Dim ser1 As New ChartSeries("Series 1")
ser1.Type = ChartSeriesType.StackingColumn
' specifying group name .
ser1.StackingGroup = "FirstGroup"
Dim ser2 As New ChartSeries("Series 2")
ser2.Type = ChartSeriesType.StackingColumn
' specifying group name .
ser2.StackingGroup = "SecondGroup"
Dim ser3 As New ChartSeries("Series 3")
ser3.Type = ChartSeriesType.StackingColumn
' specifying group name .
ser3.StackingGroup = "FirstGroup"
```

## 4.11 Realtime

Essential Chart is optimized to deal with real time data. It can work with both huge and real time data and render a smooth and dynamic chart using any of the several available chart types.

Essentially, this involves updating the chart's data points list and optionally updating the chart axis ranges if the default ranges are not user-friendly.

While you can use the **ChartSeries.Points** to add new data points to the existing list, for best performance it's recommended to implement your own "model" to store the data points in real-time scenarios.

A sample application that illustrates this, is distributed along with the Essential Chart installation and can be found at:

[Sample Location: "<sample installation location>\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Real Time\Chart Recorder"](#)

## 4.12 Statistical Formulas

Essential Chart comes with support for several Statistical formulas that will let you apply formulas on data points in one or more series and will help you analyze and arrive at meaningful information from those data points.

These formulas are mainly exposed via static methods from the **BasicStatisticalFormulas** and **UtilityFunctions** types.

### 4.12.1 Basic Statistical Formulas

Some commonly used statistical formulas that you can apply on the series points are exposed via static methods in the **BasicStatisticalFormulas** type.

Statistical Formulas	Description
<a href="#">Anova Test</a>	An ANOVA test is used to test the differences between the means of two or more groups of data.
<a href="#">Correlation</a>	The Correlation formula shows how strong the relation is between two random variables.
<a href="#">Covariance</a>	The Covariance formula measures the degree of dependence between two random variables.
<a href="#">F-Test</a>	An F-test is any statistical test in which, the test statistic has an F-distribution if the null hypothesis is <b>true</b> .
<a href="#">Mean</a>	This formula returns the average, or mean, of data stored in a data series.
<a href="#">Median</a>	This formula returns the median for data stored in a data series.
<a href="#">Standard Deviation</a>	This formula returns the Standard Deviation within a group of data.
<a href="#">T Test with Equal Variances</a>	Perform a T Test using Student's distribution (T distribution) with equal variances.
<a href="#">T Test Paired</a>	Performs a T Test using Student's distribution (T distribution) with paired samples. This is useful when there is a natural pairing of observations in

	samples. (i.e. when a sample group is tested twice.)
<a href="#">T Test with unequal variances</a>	Perform a T Test using Student's distribution (T distribution) with unequal variances.
<a href="#">Variance</a>	This formula returns the variance within a group of data.
<a href="#">Z Test</a>	This formula performs a Z Test using Normal distribution.

#### 4.12.1.1 Anova Test

**Anova** stands for **Analysis Of Variance**. It is a technique to test the hypothesis that the means among two or more groups of data are equal and thereby, testing the differences between their variances, under the assumption that the sampled groups are normally distributed.

The test actually compares the variation between the groups with the variation within the groups and produces the results based on the values of these variations. If the between variation is larger than the within variation, the means of the groups will not be equal. If both these variations are of approximately the same size, then there will not be any significant difference between the means.

##### Steps to perform an Anova test

The null hypothesis is that there is no difference between the means and the alternative hypothesis is that at least one mean is different.

The following assumptions must be satisfied before performing the test.

- The groups from which the samples were obtained must be normally distributed.
- The groups are sampled randomly.
- The samples must be independent.
- The variances of the groups must be equal.
- The null hypothesis.

1. Calculate the **Sum of Squares** for total, between and within variations.

##### Total Variation

$$SS_{\text{total}} = \left( \sum y_1^2 + \sum y_2^2 + \dots + \sum y_r^2 \right) - \frac{\left( \sum y_1 + \sum y_2 + \dots + \sum y_r \right)^2}{N}$$

### Between Variation

$$SS_{\text{among}} = \left[ \frac{(\sum y_1)^2}{n_1} + \frac{(\sum y_2)^2}{n_2} + \dots + \frac{(\sum y_r)^2}{n_r} \right] - \frac{(\sum y_1 + \sum y_2 + \dots + \sum y_r)}{N}$$

Where,

y is the individual y points of the series,

r is the number of series present,

N is the total number of y points for all the series and

n is the number of y points in each series.

### Within Variation

$$SS_{\text{within}} = SS_{\text{total}} - SS_{\text{among}}$$

2. Using the above quantities, calculate the **degrees of freedom(df)** for these variations.

### Between Variation

$$df_{\text{among}} = r - 1$$

### Within Variation

$$df_{\text{within}} = N - r$$

Where,

r is the number of series present and

N is the total number of Y points for all the series.

3. As the next step, calculate the Mean Squares of these variations. The mean square for a variation can be calculated simply by dividing its sum of square by its degrees of freedom.

### Between Variation

$$MS_{\text{among}} = \frac{SS_{\text{among}}}{df_{\text{among}}}$$

### Within Variation

$$MS_{\text{within}} = \frac{SS_{\text{within}}}{df_{\text{within}}}$$

4. Finally, calculate **F Ratio** as below and get the **F Critical Value**.

$$F = \frac{MS_{\text{among}}}{MS_{\text{within}}}$$

5. Make your decision as below.

- If the *between* variance is smaller than the *within* variance, then the *means* are really close to each other and you will fail to reject the null hypothesis.
- If the F ratio is greater than the F critical value, then the decision will be to reject the null hypothesis and thereby conclude that at least one of the means is different.

### APIs Used

Essential Chart provides support to perform Anova Test by implementing a method named Anova in the **BasicStatisticalFormulas** class. This method does the above described calculations and returns the test results as an instance of **AnovaResult** class. The AnovaResult is a class implemented to store the anova test results such as sum of squares, degrees of freedom and mean squares for different variations and also stores the **FRatio** and **FCriticalValue** of the test. Below is a detailed table for the Anova method.

Method Name	Parameters	Return Values
Anova	1. Probability: the alpha value (probability). 2. InputSeries: references to two or more input series. Each series must exist in the series collection at the time of the method call, and have the same number of data points.	An Anova has the following members: <ul style="list-style-type: none"> <li>• DegreeOfFreedomBetweenGroups</li> <li>• DegreeOfFreedomTotal</li> <li>• DegreeOfFreedomWithinGroups</li> <li>• FCriticalValue</li> <li>• FRatio</li> <li>• MeanSquareVarianceBetweenGroups</li> <li>• MeanSquareVarianceWithinGroups</li> <li>• SumOfSquaresBetweenG</li> </ul>

		roups • SumOfSquaresTotal • SumOfSquaresWithinGro ups
--	--	----------------------------------------------------------------

Here is a sample code snippet to simulate an Anova test.

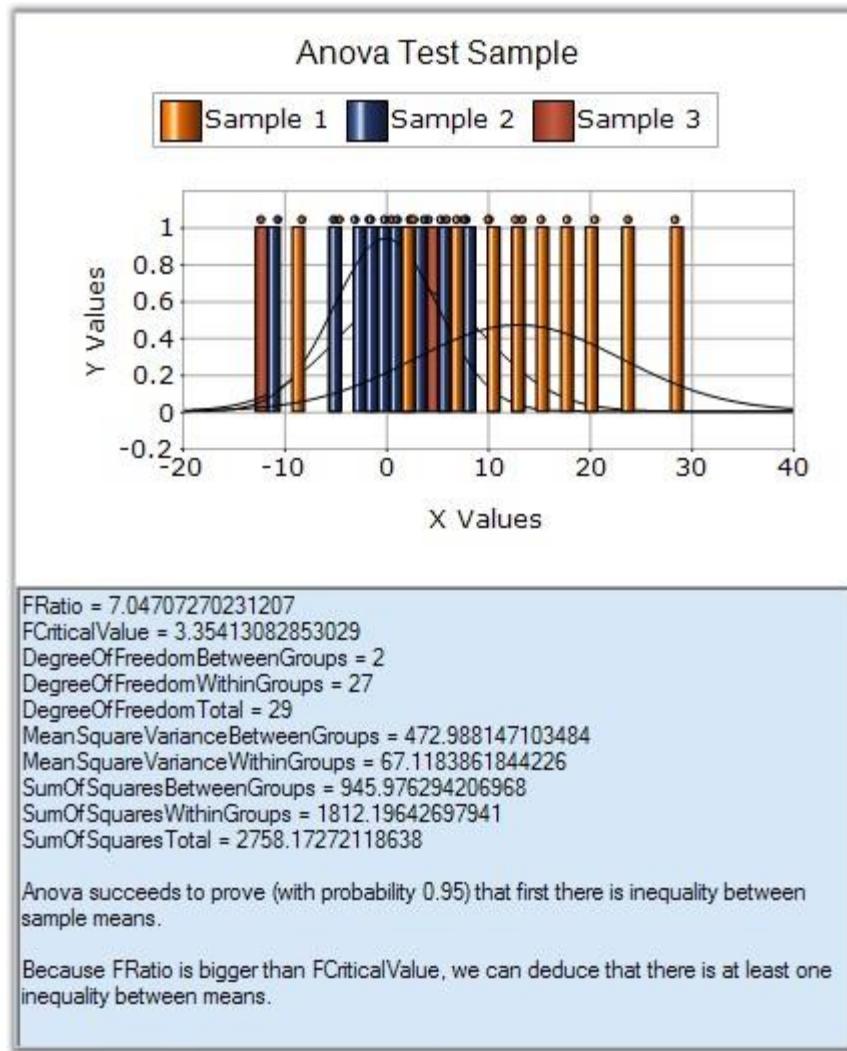
**[C#]**

```
AnovaResult ar = BasicStatisticalFormulas.Anova(confidenceLevel,  
new ChartSeries[]{ series1, series2, series3 } );
```

**[VB .NET]**

```
Dim ar As AnovaResult = BasicStatisticalFormulas.Anova(confidenceLevel,  
New ChartSeries(){ series1, series2, series3 })
```

The following image displays the results of an ANOVA test.



*Figure 346: Anova Test*

#### 4.12.1.2 Correlation

**Correlation**, which is otherwise called as **Correlation Coefficient**, is a statistical formula that determines the degree of relationship between the y values of two series representing two variables. This calculation will then be used to measure the depth of synchronization between those two variables.

A **relationship** generally refers to the correspondence between two variables. For instance, let us have two variables with one representing 'Work Experience' while the other indicating the 'Salary Expectation'. If the 'Work Experience' is high, then the 'Salary Expectation' will also be high. When 'Work Experience' is low, then the 'Salary Expectation' also tends to be low. Thereby these two variables are correlated.

The relationships can be classified into two types, **Positive** and **Negative**. In a **positive** relationship, high values on one variable are associated with high values on the other and low values on one variable are associated with low values on the other. The **negative** relationship is just the inverse of the positive relationship where the high values on one variable are associated with low values on the other.

When the measured correlation coefficient is positive, the series values would be positively correlated where as if the correlation coefficient is negative, then the series values would then be negatively correlated. Below is the formula for calculating correlation coefficient.

$$r = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{[N\sum x^2 - (\sum x)^2][N\sum y^2 - (\sum y)^2]}}$$

where,

x is the y value of first series and

y is the y value of second series.

### Using the Formula

The Correlation Coefficient can easily be calculated by using the Correlation method available with the BasicStatisticalFormulas class. The following table describes the details of this method. This method returns the covariance of the datasets divided by the product of their standard deviations.

Method Name	Parameters	Return Value
Correlation	<p><b>1. FirstInputSeries:</b> A ChartSeries object that stores the first group's data.</p> <p><b>2. SecondInputSeries:</b> A ChartSeries object that stores the second group's data. An exception will be raised if the input series does not have the same number of data points.</p>	A double that represents the correlation value between the two groups of data. The value always ranges from -1 to 1.

## Example

The below code snippet demonstrates how to get the correlation coefficient between two groups of data (Series1 and Series2) using the in-built formula.

### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
.....
double Correlation1=
BasicStatisticalFormulas.Correlation(series1,series2);
```

### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
.....
Dim Correlation1 As Double
Correlation1=BasicStatisticalFormulas.Correlation(series,series1)
```



**Note:** For further details, refer to this [Browser Sample](#):

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](http://[Installed drive]:\Documents and Settings\[User name]\My Documents\Syncfusion\EssentialStudio\[Installed version]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas)

### 4.12.1.3 Covariance

**Covariance** is a statistical formula that measures the extent to which the y values of two series vary together. It is basically used to measure the fluctuations between two quantities. For a given pairs of series y values, the covariance can be calculated by taking their differences from their mean values and multiplying these differences together. That is,

$$\text{Cov}(x,y) = \Sigma\{ [x - \bar{x}] [y - \bar{y}] \}$$

If this product is **positive**, then the values would be varying in the same direction; if it is **negative**, then the values would be varying in opposite directions. If the product is **zero**, then we can conclude that there is no linear relationship between the series values. The above formula can be simplified as below.

$$\text{Cov}(x,y) = \Sigma\{xy\} - \bar{x}\bar{y}$$

## Using the Formula

The Covariance can easily be calculated by using the **Covariance** method available with the **BasicStatisticalFormulas** class. The following table describes the details of this method.

Method Name	Parameters	Return Value
Covariance	1. <b>FirstInputSeriesName</b> : A ChartSeries object that stores the first group's data. 2. <b>SecondInputSeriesName</b> : A ChartSeries object that stores the second group's data. An exception will be raised if the input series do not have the same number of data points.	A <b>double</b> value that represents the covariance value between the two groups of data.

### Example

Here is the code snippet that demonstrates the usage of this method.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
.....
double Covariance1=
Statistics.BasicStatisticalFormulas.Covariance(series1,series2);
```

#### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
.....
Dim Covariance1 As Double
Covariance1=BasicStatisticalFormulas.Covariance (series,series1)
```



**Note:** For further details, refer to this Browser Sample:

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](http://[Installed drive]:\Documents and Settings\[User name]\My Documents\Syncfusion\EssentialStudio\[Installed version]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas)

#### 4.12.1.4 F-Test

The **F-Test** is a statistical test, which is carried out to find out whether two series have the same standard deviation with the specified confidence level. It is achieved by comparing the variances of the series values and thereby comparing their standard deviations. Here, the null hypothesis is that the two variances are equal. All hypothesis testing is done under the assumption that the null hypothesis is **true**.

##### Steps to perform F-Test

1. Calculate the variances of both the series.
2. Calculate F Ratio as given below.

$$F \text{ Value} = \text{firstSeriesVariance} / \text{secondSeriesVariance}.$$

F-Test can be easily performed by using the **FTest** method of **BasicStatisticalFormulas** class that returns the results as a type of **FTestResult**. The **FTestResult** is a class implemented to save the F test result as **FValue** and other computation results such as series means, series variances, FRatio and FCriticalValue of the test. Below is a detailed table for the **FTest** method.

Method Name	Parameters	Returns
FTest	1. <b>Probability</b> : Probability that gives the confidence level. 2. <b>FirstInputSeries</b> : Type of ChartSeries object that represents the first group of data. 3. <b>SecondInputSeries</b> : Type of ChartSeries object that represents the second group of data.	An <b>FTestResult</b> has the following members: FirstSeriesMean SecondSeriesMean FirstSeriesVariance SecondSeriesVariance FValue ProbabilityFOneTail FCriticalValueOneTail

##### Example

Here is a code snippet that shows a sample usage.

[C#]

```
FTestResult ttr =
Syncfusion.Windows.Forms.Chart.Statistics.BasicStatisticalFormulas.FTe
```

```
t(confidenceLevel,series1,series2);
```

**[VB .NET]**

```
Dim ttr As FTestResult =
Syncfusion.Windows.Forms.Chart.Statistics.BasicStatisticalFormulas.FTest(confidenceLevel,series1,series2)
```



**Note:** For further details, refer to this [Browser Sample](#):

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](#)

#### 4.12.1.5 Mean

**Mean** is statistical formula that returns the arithmetic average of series y values where the arithmetic average is the sum of all y values of a series divided by the total number of y values present in that series. The arithmetic mean can be calculated for any chart series by using **Mean** method of the **BasicStatisticalFormulas** class. Below table shows the method details.

Method Name	Parameter	Return value
Mean	<b>InputSeries:</b> A ChartSeries type object for whose y values an average is required.	A <b>double</b> that represents the average of all the y values in the given series.

#### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;
.....
double calculatedMean = BasicStatisticalFormulas.Mean(series1);
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
```

```
.....  
Dim calculatedMean As Double  
calculatedMean = BasicStatisticalFormulas.Mean(series1)
```



**Note:** For further details, refer to this [Browser Sample](#):

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](#)

#### 4.12.1.6 Median

**Median** is a statistical formula that is used to find the median of y values of a series. Median can be calculated by arranging the values from the lowest to the highest and picking up the middle one. If the total number of values is even, then pick up the two middle values after sorting the values in ascending order. The mean of these two middle values will give you the median. Hence half of the series points have values less than the median and the values of the other half will be greater than the median.

Median can be found out for any series by using the Median method of **BasicStatisticalFormulas** class. The below table shows the details of this method.

Method Name	Parameter	Return value
Median	<b>InputSeries:</b> A ChartSeries type object for whose X values an average is required.	A <b>double</b> that represents the Median value of all the X values in the given series.

#### Example

Here is a code snippet that shows a sample usage.

```
[C#]  
  
using Syncfusion.Windows.Forms.Chart.Statistics;  
.....  
double calculatedMedian =  
Statistics.BasicStatisticalFormulas.Median(series1);
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
.....
Dim Median1 As Double
calculatedMedian = BasicStatisticalFormulas.Median(series1)
```



**Note:** For further details, refer to this [Browser Sample](#):

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](http://[Installed drive]:\Documents and Settings\[User name]\My Documents\Syncfusion\EssentialStudio\[Installed version]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas)

#### 4.12.1.7 Standard Deviation

**StandardDeviation** is the statistical formula that is basically used to measure the variability. That is, it can be used to measure how spread out your data are. It can be defined as the square root of the variance where a variance is the average of the squared differences between the data points and the mean. In other words, it is named as the 'root-mean-square' of the data values.

It can be used to check how tightly the data values are clustered around the mean. If the data points are close to the mean, then the standard deviation will be small or if the points are far from the mean, then the standard deviation is large or if all the data values are equal, then the standard deviation is **zero**.

The Standard Deviation can be calculated for any series by using the **StandardDeviation** method of **BasicStatisticalFormulas** class. Below is the detailed description of this method.

Method Name	Parameters	Return Value
StandardDeviation	1. <b>InputSeries</b> : A ChartSeries type object for on whose Y values this formula should be applied. 2. <b>SampleVariance</b> : true if the data is a sample of a population, false if it is the entire population.	A <b>double</b> that represents the standard deviation within the group of data.

#### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;  
.....  
double Deviation1 =  
BasicStatisticalFormulas.StandartDeviation(series1, false);
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
.....  
Dim Deviation1 As Double  
Deviation1 = BasicStatisticalFormulas. StandartDeviation  
(series1, false)
```



**Note:** For further details, refer to this Browser Sample:

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](http://[Installed drive]:\Documents and Settings\[User name]\My Documents\Syncfusion\EssentialStudio\[Installed version]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas)

#### 4.12.1.8 T-Tests

TTest is a statistical formula that is used to measure the equality between the means of two series. In other words, the T test compares the actual difference between two means in relation to the variation in data that can be measured by calculating the standard deviation of the difference between the means.

It is a statistical test used to test the null hypothesis that the means of two normally distributed populations are equal. This test can be performed on two given samples(series), each characterized by its mean, standard deviation and number of data points, to determine whether the means are distinct based on the assumption that the underlying distributions are normal.

##### Different T-Tests

There are different versions of T tests depending on whether the samples are:

- **independent** of each other, (where the series are random with no relationship between each other)

**OR**

- **paired**, (where every data point in one series will have a relationship with a particular point of another series).

If the calculated t-statistic exceeds the chosen threshold value (usually 0.05), then the decision is to reject the null hypothesis, which states that the two sample means are equal, in favor of an alternate hypothesis, which typically specifies that the two samples differ.

Below are the different formulae to calculate the t-statistic:

- **T-Test with Equal Variances**

This formula performs a T test for two groups of data and assumes equal variances between the two groups (i.e. series).

- **T-Test Paired**

This formula performs a paired two-sample student's t-test to determine whether a sample's means are distinct. This form of the t-test does not assume that the variances of both the populations are equal.

Use a paired test when there is a natural pairing of observations in the samples, such as a sample group that is tested twice. (e.g. before and after an experiment)

- **T-Test with UnEqual Variances**

This formula performs a T-test for two groups of data and assumes unequal variances between the two groups. (i.e. series)

This analysis tool is referred to as a heteroscedastic t-test and can be used when the groups that are under study are distinct. Use a paired test when there is one group before and after a treatment.

***Note: For programming example, refer to the following Browser Sample:***

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\Installed version\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](file:///C:/Users/My%20Documents/Syncfusion/EssentialStudio/Installed%20version/Windows/Chart.Windows/Samples/2.0/Statistical%20Analysis/Chart%20Statistical%20Formulas)

#### **4.12.1.8.1 TTest with Equal Variances**

This type of TTest can be performed on two random series that have no relationship with each other. But they should be of equal sizes, i.e., the number of data points of the two series should be same.

### Steps to perform the test

1. Specify the null hypothesis and alternate hypothesis.
  - Null Hypothesis - Difference between the two means is **zero**.
  - Alternate Hypothesis - Difference between the two means is not **zero**.
2. Calculate the means of the two input series ( $\mu_1$  and  $\mu_2$ ) and calculate their difference ( $Md$ ).

$$Md = \mu_1 - \mu_2$$

3. Calculate the variances of the two input series ( $s_1$  and  $s_2$ ).
4. Let  $n_1$  and  $n_2$  be the number of data points in first and second series respectively.
5. Calculate the degrees of freedom.

$$D = n_1 + n_2 - 2$$

6. As a next step, Calculate the Pooled Estimator as below.

$$Sp = (n_1 - 1) * s_1 + (n_2 - 1) * s_2$$

7. Calculate the T-statistic as given below.

$$t = (\mu_1 - \mu_2 - Md) / \sqrt{Sp/n_1 + Sp/n_2}$$

8. Construct a t-table at ( $n_1+n_2-2$ ) degrees of freedom.
9. Choose a level of significance(probability), say  $p = 0.05$  and read the tabulated value.
10. If the calculated tvalue exceeds the tabulated value we can say that the means are significantly different at that level of probability.

### Using the Formula

The TTest formula for equal variances can be calculated by using the **TTestEqualVariances** method of the **BasicStatisticalFormulas** class. The following table presents the details of this method. This method returns an instance of **TTestResult** class that stores the resultant values of this test such as means of the two series, T value, degrees of freedom, number of points in every series, T critical value and confidence level(probability).

Method Name	Parameters	Return Value
TTestEqualVariances	1. <b>HypothesizedMeanDifference</b> : A <b>double</b> value specifying the difference between two population means.	A <b>TTestResult</b> object that has the following members:

	<p>2. <b>Probability:</b> A <b>double</b> value that gives the confidence level.</p> <p>3. <b>FirstInputSeries:</b> A <b>ChartSeries</b> object that stores the first group of data.</p> <p>4. <b>SecondInputSeries:</b> A <b>ChartSeries</b> object that stores the second group of data.. .</p>	<ul style="list-style-type: none"> <li>• FirstSeriesMean</li> <li>• SecondSeriesMean</li> <li>• FirstSeriesVariance</li> <li>• SecondSeriesVariance</li> <li>• Tvalue</li> <li>• DegreeOfFreedom</li> <li>• ProbabilityTOneTail</li> <li>• TCriticalValueOneTail</li> <li>• ProbabilityTTwoTail</li> <li>• TCriticalValueTwoTail</li> </ul>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
TTestResult ttr = BasicStatisticalFormulas.TTestEqualVariances (0.2,
0.05, series1, series2);
```

#### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
Dim ttr As TTestResult = BasicStatisticalFormulas.TTestEqualVariances
(0.2, 0.05, series1, series2)
```

#### 4.12.1.8.2 T-Test with UnEqual Variance

If the assumption of 'equal variances' is violated, then we have to compute the test statistic using the individual sample's standard deviation instead of pooled standard deviation. Like the **TTestEqualVariances** formula, the **TTestUnequalVariances** formula also will be carried out on two independent samples. The only difference with unequal variances test is that the sample should be of different sizes.

#### Steps to perform the test

1. Specify the null hypothesis and alternate hypothesis.

- Null Hypothesis - Difference between the two means is **zero**.
  - Alternate Hypothesis - Difference between the two means is not **zero**.
2. Calculate the means of the two input series ( $\mu_1$  and  $\mu_2$ ) and calculate their difference ( $M_d$ ).

$$M_d = \mu_1 - \mu_2$$

3. Calculate the variances of the two input series ( $s_1$  and  $s_2$ ).
4. Let  $n_1$  and  $n_2$  be the number of data points in first and second series respectively.
5. Calculate the degrees of freedom.

$$df = \frac{\left[ \frac{s_1}{n_1} + \frac{s_2}{n_2} \right]^2}{\frac{(s_1/n_1)^2}{n_1-1} + \frac{(s_2/n_2)^2}{n_2-1}}$$

6. Calculate the T-statistic as given below.

$$t = (\mu_1 - \mu_2 - M_d) / \text{Sqrt}(s_1/n_1 + s_2/n_2)$$

7. Choose a level of significance(probability), say  $p = 0.05$  and read the tabulated value.
8. If the calculated tvalue exceeds the tabulated value we can say that the means are significantly different at that level of probability.

### Using the Formula

The TTest formula for unequal variances can be calculated by using the TTestUnEqualVariances method of the BasicStatisticalFormulas class. The following table presents the details of this method. This method returns an instance of TTestResult class that stores the resultant values of this test such as means of the two series, T value, degrees of freedom, number of points in every series, T critical value and confidence level(probability).

Method Name	Parameters	Return Value
TTestUnEqualVariances	1. <b>HypothesizedMeanDifference:</b> A <b>double</b> value that gives the difference between the means	A TTestResult object that has the following members: <ul style="list-style-type: none"> <li>• FirstSeriesMean</li> </ul>

	<p>of the two input series.</p> <p><b>2. Probability:</b> A <b>double</b> value that denotes the probability that gives the confidence level.</p> <p><b>3. FirstSeries:</b> A <b>ChartSeries</b> object that stores the first group of data.</p> <p><b>4. SecondSeries:</b> A <b>ChartSeries</b> object that stores the second group of data.</p>	<ul style="list-style-type: none"> <li>• SecondSeriesMean</li> <li>• FirstSeriesVariance</li> <li>• SecondSeriesVariance</li> <li>• Tvalue</li> <li>• DegreeOfFreedom</li> <li>• ProbabilityTOneTail</li> <li>• TCriticalValueOneTail</li> <li>• ProbabilityTTwoTail</li> </ul> TCriticalValueTwoTail
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
TTestResult ttr = BasicStatisticalFormulas.TTestUnEqualVariances(0.2,
0.05,series1,series2);
```

#### [VB .NET]

```
Dim ttr As TTestResult =
BasicStatisticalFormulas.TTestUnEqualVariances(0.2, 0.05, series1,
series2)
```

#### 4.12.1.8.3 TTest Paired

This formula is used when there is a dependency between the samples. The two possible scenarios could be when there is a single sample that is tested twice (before and after an experiment), or when there are two samples whose values can be matched. This test assumes that there is some difference between the means of two input series populations. Input series are regarded as samples from normally distributed populations. The population variances are assumed to be unequal. This test is otherwise called as **Robust TTest**.

#### Steps to perform the test

1. 1. Specify the null hypothesis and alternate hypothesis.
  - Null Hypothesis: Difference between the two means is zero.
  - Alternate Hypothesis: Difference between the two means is not zero.

2. Calculate the difference between two series on each pair of values.

Calculate the mean difference (**Mdiff**), ie., mean of the new series values.

3. Calculate the Standard Deviation of the differences(**Sd** ).
4. Get the degrees of freedom.

$$df = n_1 - 1$$

5. Compute the t-statistic as given below.

$$t = (M_{\text{diff}} - M_d) / [S_d * \text{Sqrt}(1/n_1)]$$

6. Construct a t-table at ( $n_1 - 1$ ) degrees of freedom and get the tabulated value for a given level of significance(probability).
7. If the calculated tvalue exceeds the tabulated value we can say that the means are significantly different at that level of probability.

### Using the Formula

The TTest formula for dependent samples can be calculated by using the **TTestPaired** method of the **BasicStatisticalFormulas** class. The following table presents the details of this method. This method returns an instance of **TTestResult** class that stores the resultant values of this test such as means of the two series, T value, degrees of freedom, number of points in every series, T critical value and confidence level(probability).

Method Name	Parameters	Return Value
TTestPaired	<ol style="list-style-type: none"> <li>1. <b>HypothesizedMeanDifference:</b> A <b>double</b> value specifying the difference between two population means.</li> <li>2. <b>Probability:</b> A <b>double</b> value that denotes the probability that gives the confidence level.</li> <li>3. <b>FirstSeries:</b> A <b>ChartSeries</b> object that stores the first group of data.</li> <li>4. <b>SecondSeries:</b> A <b>ChartSeries</b> object that stores the second group</li> </ol>	<p>A <b>TTestResult</b> object that has the following members:</p> <ul style="list-style-type: none"> <li>• <b>FirstSeriesMean</b></li> <li>• <b>SecondSeriesMean</b></li> <li>• <b>FirstSeriesVariance</b></li> <li>• <b>SecondSeriesVariance</b></li> <li>• <b>Tvalue</b></li> <li>• <b>DegreeOfFreedom</b></li> </ul>

	of data.	<ul style="list-style-type: none"><li>• ProbabilityTOneTail</li><li>• TCriticalValueOneTail</li><li>• ProbabilityTTwoTail</li><li>• TCriticalValueTwoTail</li></ul>
--	----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

### **Example**

Here is a code snippet that shows a sample usage.

#### **[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;

TTestResult ttr = BasicStatisticalFormulas.TTestPaired(0.2, 0.05,
series1, series2);
```

#### **[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics

Dim ttr As TTestResult = BasicStatisticalFormulas.TTestPaired(0.2,
0.05, series1, series2)
```

### **4.12.1.9 Variance**

**Variance** is a statistical formula that calculates the variance of series y values. A Variance can be defined as the square of the standard deviation of a sample.

#### **Using the Formula**

The variance can be computed for any series by using the method **Variance** of **BasicStatisticalFormulas** class. Below table shows the details of this method.

Method Name	Parameters	Return Value
Variance	1. <b>InputSeries</b> : A ChartSeries object that represents the input series. 2. <b>SampleVariance</b> : A boolean value; <b>true</b> if the data is a sample of a population, <b>false</b> if it is the entire population.	A <b>double</b> that represents the variance within the group of data.

### Example

Variance is the square of the standard deviation for the given data.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
.....
double Variance1= BasicStatisticalFormulas.Variance(series1,false);
```

#### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
.....
Dim Variance1 As Double
Variance1=BasicStatisticalFormulas.Variance (series1,false)
```

### 4.12.1.10 Z-Test

**Z-test** is a statistical formula that is used to determine if the difference between a sample mean and the population mean is large enough to be statistically significant. This test is primarily used to determine if the test scores of the samples are either within or outside the standard scores.

#### Steps to perform ZTest

This test requires the sample to be random and is taken from a population that is distributed normally. In order to perform this test, the following quantities should be known.

- s (the standard deviation of the population)
- $\mu$  (the mean of the population)
- x (the mean of the sample)
- n (the size of the sample)

1. Calculate the standard error (SE) of the mean:

$$SE = \frac{\sigma}{\sqrt{n}}$$

2. Then compute the z-score for the Z-test as below.

$$z = \frac{x - \mu}{SE}$$

3. Finally, the z score is compared to a Z table which contains the percent of area under the normal curve between the mean and the z score. Using this table will indicate whether the calculated z score is within the realm of chance or it is so different from the mean that the sample mean is unlikely to have happened by chance.

### Using the Formula

The Z-test can be carried out on any two series values by using the **ZTest** method of **BasicStatisticalFormulas** class. Below table gives the detailed description of this method. The method returns an instance of **ZTestResult** object that saves the intermediate results and also the final z score of the test.

Method Name	Parameters	Return Value
ZTest	<p>1. <b>HypothesizedMeanDifference</b>: the difference between the population means.</p> <p>2. <b>VarianceOfFirstSeries</b>: the variance within the first series population.</p> <p>3. <b>VarianceOfSecondSeries</b>: the variance within the second series population.</p> <p>4. <b>Probability</b>: the probability that gives the confidence level.</p> <p>5. <b>FirstSeries</b>: A ChartSeries object that stores the first group of data.</p> <p>6. <b>SecondSeries</b>: A ChartSeries object that stores the second group of data.</p>	<p>An <b>ZTestResult</b> object that has the following members:</p> <ul style="list-style-type: none"> <li>• FirstSeriesMean</li> <li>• SecondSeriesMean</li> <li>• FirstSeriesVariance</li> <li>• SecondSeriesVariance</li> <li>• ZValue</li> <li>• ProbabilityZOneTail</li> <li>• ZCriticalValueOneTail</li> <li>• ProbabilityZTwoTail</li> <li>• ZCriticalValueTwoTail</li> </ul>

### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
ZTestResult ztr = BasicStatisticalFormulas.ZTest(
Convert.ToDouble(TextBox6.Text.ToString()),
sqrtVarianceOfFirstSeries*sqrtVarianceOfFirstSeries,
sqrtVarianceOfSecondSeries*
sqrtVarianceOfSecondSeries, 0.05, series1, series2);
```

**[VB.NET]**

```
Dim ztr As ZTestResult =
BasicStatisticalFormulas.ZTest(Convert.ToDouble(TextBox6.Text.ToString(
)), sqrtVarianceOfFirstSeries*sqrtVarianceOfFirstSeries,
sqrtVarianceOfSecondSeries*sqrtVarianceOfSecondSeries, 0.05, series1,
series2)
```



**Note:** For programming example, refer to the following Sample:

[\[Installed drive\]:\Documents and Settings\\[User name\]\My Documents\Syncfusion\EssentialStudio\\[Installed version\]\Windows\Chart.Windows\Samples\2.0\Statistical Analysis\Chart Statistical Formulas](#)

## 4.12.2 Utility Functions

Listed below are some common statistical formulas that are implemented in the **Utilities** type.

Statistical Formulas	Description
<a href="#">Beta Function</a>	The BetaFunction method returns the beta function for a given value.
<a href="#">Beta Cumulative Distribution</a>	The Beta Cumulative Distribution method returns the Beta cumulative distribution for a given value.
<a href="#">Inverse Beta Cumulative Distribution</a>	The Inverse Beta Cumulative Distribution method returns the Inverse Beta cumulative distribution for a given value.

<a href="#">Gamma Function</a>	The GammaFunction method returns the gamma function for a given value.
<a href="#">Gamma Cumulative Distribution</a>	The Gamma Cumulative Distribution method returns the Gamma cumulative distribution for a given value.
<a href="#">FCumulative Distribution</a>	The F Cumulative Distribution method returns the FCumulative distribution for a given value.
<a href="#">Inverse FCumulative Distribution</a>	The Inverse F Cumulative Distribution method returns the Inverse F Cumulative distribution for a given value.
<a href="#">TCumulative Distribution</a>	The T Cumulative Distribution method returns the T Cumulative distribution for a given value.
<a href="#">Inverse TCumulative Distribution</a>	The Inverse T Cumulative Distribution method returns the Inverse T Cumulative distribution for a given value.
<a href="#">Normal Distribution</a>	The Normal Distribution method returns the Normal distribution for a given value.
<a href="#">Inverse Normal Distribution</a>	Returns the inverse of the standard normal cumulative distribution. The distribution has a mean of 0 (zero) and a standard deviation of one.
<a href="#">Normal Distribution Density</a>	The Normal Distribution Density method returns the Normal distribution density for a given value.
<a href="#">Binomial</a>	The Binomial method returns the Binomial coefficient for a given Value.
<a href="#">Factorial</a>	The Factorial method returns the Factorial for a given Value.(eg $2!=2$ ).
<a href="#">Error Function</a>	The Erf method returns the Error Function for a given Value.
<a href="#">Inverse Error Function</a>	The Inverse Erf method returns the Inverse Error Function for a given Value.

#### 4.12.2.1 Beta Function

There are two widely used utility functions, the Gamma and Beta functions, which are used in statistics to calculate distribution values. These functions always return a double value and use two double values for input. The beta function was studied by Euler and Legendre and was named by Jacques Binet. In mathematics, the beta function (occasionally written as Beta function) which, is also called the Euler integral of the first kind, is a special function defined by:

$$B(x, y) = \frac{\Gamma(x) \Gamma(y)}{\Gamma(x + y)}$$

where  $\Gamma(x)$  is the gamma function.

### Using the Formula

The **Beta** method of the **UtilityFunctions** class calculates the beta function for given two values.

Method Name	Parameters	Return Value
Beta	1. a: The first value. 2. b: The second value.	A double that represents the beta function value.

### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;  
double result = UtilityFunctions.Beta(a,b);
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
Dim double as result = UtilityFunctions.Beta(a,b);
```

### 4.12.2.2 Beta Cumulative Distribution

The **Beta Distribution** can be defined as a family of probability distributions differing in the values of  $\alpha$  and  $\beta$ . The **Cumulative distribution** function is given below.

$$F(x; \alpha, \beta) = \frac{B_x(\alpha, \beta)}{B(\alpha, \beta)}$$

where  $B_x(\alpha, \beta)$  is the incomplete beta function and  $I_x(\alpha, \beta)$  is the regularized incomplete beta function.

### Using the Formula

The **BetaCumulativeDistribution** method of the **UtilityFunctions** class returns the cumulative beta distribution for  $x \geq 0$ ,  $a > 0$ ,  $b > 0$ .

Method Name	Parameters	Return Value
BetaCumulativeDistribution	1. a: The lower limit. 2. b: The upper limit. 3. x: the value for which the distribution has to be calculated.	A double that represents the cumulative beta distribution.

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
ChartSeries series = this.ChartControl1.Model.NewSeries("a=b=0.5");
for(double i=0;i<=1;i=i+0.1)
{
    // Calculate Beta cumulative function for a points and plot the
    // points in chart control.
    series.Points.Add(i, Syncfusion.Windows.Forms.Chart.Statistics.UtilityFunctions.BetaCumulativeDistribution(0.5, 0.5, i));
}
series.Type = ChartSeriesType.Spline;
series.Tex = series.Name;
this.ChartControl1.Series.Add(series);
```

#### [VB .NET]

```
'Calculate Beta cumulative function for a points and plot the points in
chart control.
Dim series As ChartSeries = Me.ChartControl1.Model.NewSeries("a=b=0.5")
For i As Double = 0 To 1 Step 0.1
```

```

series.Points.Add(i, Syncfusion.Windows.Forms.Chart.Statistics.UtilityFunctions.BetaCumulativeDistribution(0.5, 0.5, i))
Next i
series.Type = ChartSeriesType.Spline
series.Text = series.Name
Me.ChartControl1.Series.Add(series)

```

### 4.12.2.3 Binomial Coefficient

**Binomial Coefficient** is an utility function used in statistical calculations. This function is used to determine the possible number of combinations of 'k' items that can be selected from a set of 'n' items. The binomial coefficient formula can be explicitly stated as given below.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{if } n \geq k \geq 0$$

where  $n!$  denotes the factorial of  $n$ .

An alternative name for the binomial coefficient is choose function; the binomial coefficient of  $n$  and  $k$  is often read as "n choose  $k$ ". Alternative notations include  $C(n, k)$ ,  $nCk$  ( $C$  for combination). These numbers are called binomial coefficients because they are coefficients in binomial theorem.

#### Using the formula

The **Binomial** method of the **UtilityFunctions** class returns the binomial coefficient for given **n** and **k** values.

Method Name	Parameters	Return Value
Binomial	1. The $n$ value. 2. The $k$ value.	An integer that represents the binomial coefficient value.

#### Example

Here is a code snippet that shows a sample usage.

```

[C#]
using Syncfusion.Windows.Forms.Chart.Statistics;
int result = UtilityFunctions.Binomial(n, k);

```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
Dim int as result = UtilityFunctions.Binomial(n, k)
```

#### 4.12.2.4 Inverse Beta Cumulative Distribution

This formula returns the inverse of Beta Cumulative Distribution.

##### Using the formula

The **InverseBetaCumulativeDistribution** method of the **UtilityFunctions** class returns the inverse of beta cumulative distribution ( for  $1 \geq p \geq 0$  ,  $a > 0$ ,  $b > 0$  ).

Method Name	Parameters	Return Value
InverseBetaCumulativeDistribution	1. a: First Parameter of Beta function. 2. b: Second Parameter of Beta function. 3. p: The probability.	A double that inverses the beta cumulative distribution value.

##### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;  
double result = UtilityFunctions.InverseBetaCumulativeDistribution (a,  
b, p);
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
Dim double as result =  
UtilityFunctions.InverseBetaCumulativeDistribution (a, b, p)
```

#### 4.12.2.5 Error Function

The Error function, denoted as  $\text{Erf}(x)$ , gives the probability that a measurement under the influence of accidental errors has a distance less than  $x$  from the average value at the center. It is the integral of Gauss curve, that is usually normalized to one with a factor of  $2/\sqrt{\pi}$ . It is otherwise called as integrated Gauss function or Gauss Error function.

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

Here is the plot of error function.

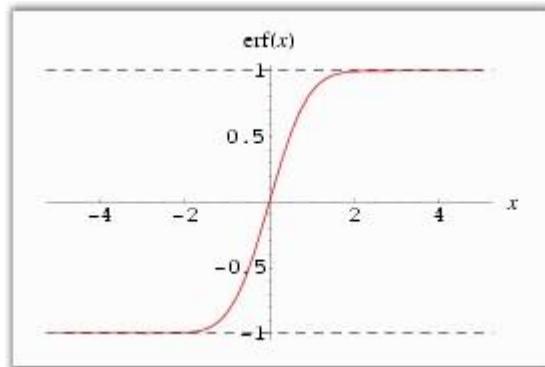


Figure 347: Error Function

#### Using the formula

The **Erf** method of the **UtilityFunctions** class returns integral of the Gauss curve for  $x > 0$ .

Method Name	Parameters	Return Value
Erf	x: must be greater than zero.	A double that represents the Gauss integral.

#### Example

Here is a code snippet that shows a sample usage.

```
[C#]
using Syncfusion.Windows.Forms.Chart.Statistics;
```

```
int double = UtilityFunctions.Erf(x);
```

**[VB.NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
Dim double as result = UtilityFunctions.Erf(x)
```

#### 4.12.2.6 Factorial

The **Factorial** function returns the factorial value of a given number. In mathematics, the factorial of a natural number  $n$  is the product of all positive integers less than or equal to  $n$ . It is denoted as  $n!$  and pronounced "n factorial", or colloquially "n shriek", "n bang" or "n crit". Factorial finds its main application in combinatorics like Permutations and Combinations and is also used in Number Theory.

The factorial function is defined by the following expression.

$$n! = \prod_{k=1}^n k \quad \forall n \in \mathbb{N}.$$

which is equivalent to  $n! = n \cdot (n-1) \cdot \dots \cdot 2 \cdot 1$ .

The above definition incorporates the convention that the product of no numbers at all is 1, i.e.,  $0! = 1$ .

#### Using the formula

The **Factorial** method of the **UtilityFunctions** class returns the factorial value for any positive integer.

Method Name	Parameters	Example
<b>Factorial</b>	n: The number whose factorial should be found out.	An integer that returns the factorial of n.

#### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;
int result = UtilityFunctions.Factorial(int n);
```

**[VB.NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
Dim int as result = UtilityFunctions.Factorial(int n)
```

#### 4.12.2.7 F Cumulative Distribution

This formula returns cumulative F Distribution which can be defined as the ratio of two chi-square distributions. The formula can be expressed as given below.

$$\frac{U_1/d_1}{U_2/d_2}$$

where,

**U1** is the first chi square distribution with d1 degrees of freedom and

**U2** is the second chi square distribution with d2 degrees of freedom.

#### Using the Formula

FCumulativeDistribution is calculated using the **Statistics.UtilityFunctions** class. The following table describes the F Cumulative distribution method.

Method Name	Parameters	Return Value
FCumulativeDistribution	1. <b>fValue</b> : The F value for which you want the distribution. 2. <b>firstDegreeOfFreedom</b> : an integer value that represents the first degree of freedom. 3. <b>secondDegreeOfFreedom</b> : an integer value that represents the second degree of freedom.	A double that represents T cumulative distribution.

#### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double x = Statistics.UtilityFunctions.FCumulativeDistribution(
fvalue, firstdegreeOfFreedom, secondDegreeOfFreedom );
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
double x = Statistics.UtilityFunctions.
FCumulativeDistribution(fvalue, firstdegreeOfFreedom,
secondDegreeOfFreedom)
```

#### 4.12.2.8 Gamma Function

The Gamma Function is an attempt to generalize the [factorial](#) function to real and complex numbers. It is related to the factorial function by

$$\Gamma(n) = (n - 1)!$$

##### **The Gamma Function**

For a complex number  $x$  with a positive real part, the function can be given by

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt, \quad x > 0$$

##### **Special Values of gamma function**

$$\Gamma(-n) = \infty$$

$$\Gamma(0) = \infty$$

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

$$\Gamma(1) = 1$$

$$\Gamma\left(\frac{3}{2}\right) = \frac{\sqrt{\pi}}{2}$$

$$\Gamma(2) = 1$$

### Using the Formula

The Gamma function is calculated using the **Statistics.UtilityFunctions** class. The following table describes the parameters and the return value of the gamma function.

Method Name	Parameters	Return Value
Gamma	p: a value for which the gamma value is required.	A double that represents the gamma function value.

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double x = Statistics.UtilityFunctions.Gamma( p );
```

#### [VB.NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
double x = Statistics.UtilityFunctions.Gamma( p )
```

#### 4.12.2.9 Gamma Cumulative Distribution

The formula for the cumulative distribution function for the gamma distribution is,

$$F(x) = \frac{\Gamma_x(\gamma)}{\Gamma(\gamma)} \quad x \geq 0; \gamma > 0$$

where  $\Gamma$  is the gamma function defined above and  $\Gamma_x$  is the incomplete gamma function. The incomplete gamma function has the formula.

$$\Gamma_x(a) = \int_0^x t^{a-1} e^{-t} dt$$

#### Example

Here is a code snippet that shows a sample usage.

##### [C#]

```
ChartSeries series=this.ChartControll1.Model.NewSeries("a=2");
for(double i=0;i<=20;i=i+2)
{
// Calculate Gamma Cumulative function for a points and plot the points
in chart control.
series.Points.Add
(i,Syncfusion.Windows.Forms.Chart.Statistics.UtilityFunctions.GammaCumulativeDistribution(2.0,i));
}
series.Type=ChartSeriesType.Spline;
series.Text=series.Name;
this.ChartControll1.Series.Add(series);
```

##### [VB .NET]

```
Dim series As ChartSeries=Me.ChartControll1.Model.NewSeries("a=2")
For i As Double = 0 To 20 Step 2

' Calculate Gamma Cumulative function for a points and plot the points
in chart control.
series.Points.Add(i,Syncfusion.Windows.Forms.Chart.Statistics.UtilityFunctions.GammaCumulativeDistribution(2.0,i))
Next i
```

```
series.Type=ChartSeriesType.Spline  
series.Text=series.Name  
Me.ChartControll1.Series.Add(series)
```

#### 4.12.2.10 Inverse Error Function

The Inverse Error function, which is a rational approximation of the error function, gives the element-by-element inverse of the error function. The absolute value of the relative error is less than 1.15 -10.9 in the entire region.

##### Using the formula

The below table describes this function in detail.

Method Name	Parameters	Return Value
InverseErrorFunction	x must be less than 1.15-10.9	A double that gives the inverse of error function.

##### Example

Here is a code snippet that shows a sample usage.

###### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;  
int double = UtilityFunctions.InverseErf(x);
```

###### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
Dim double as result = UtilityFunctions.InverseErf(x)
```

#### 4.12.2.11 Inverse F Cumulative Distribution

This formula returns the inverse of the F cumulative distribution.

##### Using the Formula

InverseFCumulativeDistribution is calculated using the **Statistics.UtilityFunctions** class. The following table describes its parameters and its values.

Method Name	Parameters	Return Value
InverseFCumulativeDistribution	1. <b>fValue</b> : The F value for which you need the distribution. 2. <b>firstDegreeOfFreedom</b> : an integer value that represents the first degree of freedom. 3. <b>secondDegreeOfFreedom</b> : an integer value that represents the second degree of freedom.	A double that represents the inverse F cumulative distribution.

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double x= Statistics.UtilityFunctions. InverseFCumulativeDistribution(
fvalue, firstdegreeOf Freedom, secondDegreeOfFreedom );
```

#### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
double x= Statistics.UtilityFunctions.
InverseFCumulativeDistribution(fvalue, firstdegreeOf Freedom,
secondDegreeOfFreedom)
```

### 4.12.2.12 Inverse Normal Distribution

This formula returns an approximation of the inverse of the standard normal cumulative distribution. That is, for a given P, it returns an approximation to the x satisfying  $P = \Pr\{z \text{ is smaller than } x\}$  where z is a random variable from the standard normal distribution.

#### Using the Formula

InverseNormalDistribution is calculated using the **Statistics.UtilityFunctions** class. The following table describes its parameters and its values.

Method Name	Parameters	Example
InverseNormalDistribution	p: the probability at which the function value is evaluated. p must be in (0,1) range.	A double that represents the inverse of the normal distribution function.

The algorithm uses a minimax approximation by rational functions and the result has a relative error whose absolute value is less than 1.15e-9.

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double x = Statistics.UtilityFunctions.InverseNormalDistribution( p );
```

#### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
double x = Statistics.UtilityFunctions.InverseNormalDistribution( p )
```

### 4.12.2.13 Normal Distribution

This formula yields the value of the standard normal cumulative distribution. Normal distributions are symmetric and have bell-shaped density curves with a single peak. Two factors, the mean ( $\mu$ ) and the standard deviation ( $\sigma$ ), come into place when we speak of normal distribution. The mean indicates the peak of the density curve and the standard deviation indicates the spread of the bell curve.

The normal density function is given by,

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{x-\mu}{\sigma}\right)^2}$$

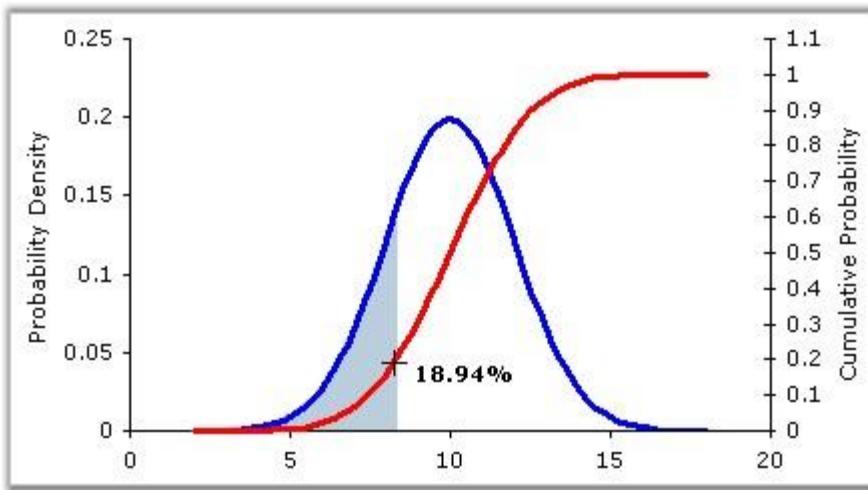


Figure 348: Normal Density Function

Different values of  $\mu$  and  $\sigma$  yield different normal density curves and hence different normal distributions. All normal density curves satisfy the following property which is often referred to as the Empirical Rule.

- 68% of the observations fall within 1 standard deviation of the mean, that is, between  $\mu - \sigma$  and  $\mu + \sigma$ .
- 95% of the observations fall within 2 standard deviations of the mean, that is, between  $\mu - 2\sigma$  and  $\mu + 2\sigma$ .
- 99.7% of the observations fall within 3 standard deviations of the mean, that is, between  $\mu - 3\sigma$  and  $\mu + 3\sigma$ .

Thus, for a normal distribution, almost all values lie within three standard deviations of the mean.

### Using the Formula

NormalDistribution is calculated using the **Statistics.UtilityFunctions** class. The following table describes this formula's parameters and its values.

Method Name	Parameters	Return Value
NormalDistribution	zValue: The value for which the distribution is required.	A double that represents the standard normal cumulative distribution value.

### Example

Here is a code snippet that shows a sample usage.

**[C#]**

```
using Syncfusion.Windows.Forms.Chart.Statistics;  
double x = Statistics.UtilityFunctions.NormalDistribution( p );
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
double x = Statistics.UtilityFunctions.NormalDistribution( p )
```

#### 4.12.2.14 Normal Distribution Density

In probability and statistics, the log-normal distribution is the probability of distribution of any random variable whose logarithm is normally distributed (the base of the logarithmic function is immaterial in that  $\log_a x$  is normally distributed if and only if  $\log_b X$  is normally distributed). If  $x$  is a random variable with a normal distribution, then  $\exp(X)$  will have a log-normal distribution.

"Log-normal" can also be written as "log normal", "lognormal" or "logistic normal".

A variable might be modeled as log-normal if it can be thought of as the multiplicative product of many small independent factors. A typical example of this is the long-term return rate on a stock investment: it can be considered as the product of the daily return rates.

The log-normal distribution has a probability density function (pdf),

$$f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln x - \mu)^2/2\sigma^2}$$

for  $x > 0$ , where  $\mu$  and  $\sigma$  are the median and standard deviation of the variable's logarithm. The expected value is,

$$E(X) = e^{\mu + \sigma^2/2}$$

and the variance is,

$$\text{var}(X) = (e^{\sigma^2} - 1)e^{2\mu + \sigma^2}$$

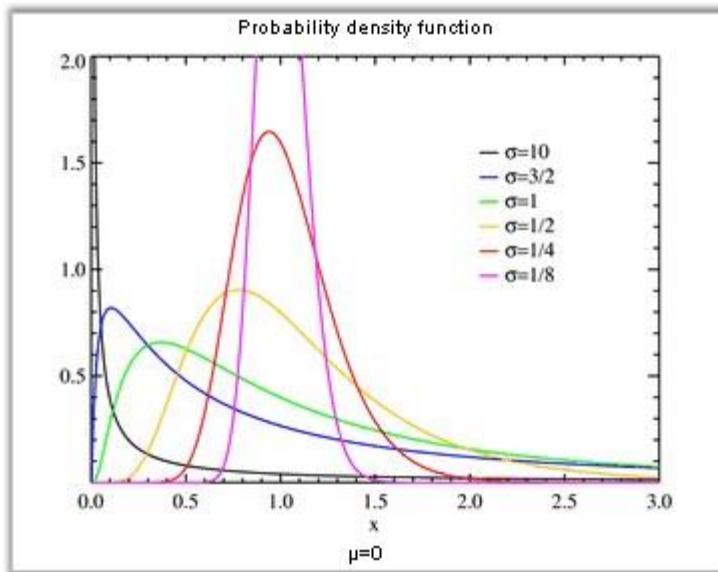


Figure 349: Normal Distribution Density

### Using the formula

Method Name	Parameters	Return Value
NormalDistributionDensity	1. <b>x</b> : the value at which the distribution density is evaluated. 2. <b>m</b> : the expected value of distribution. 3. <b>sigma</b> : the variance of distribution.	A double that represents the Normal Distribution Density function value.

### Example

Here is a code snippet that shows a sample usage.

#### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double result = UtilityFunctions.NormalDistributionDensity(x, m
,sigma);
```

#### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
Dim double as result = UtilityFunctions.NormalDistributionDensity(x, m
,sigma)
```

### 4.12.2.15 Inverse T Cumulative Distribution

This formula computes the inverse of the cumulative distribution for T-statistic.

#### Using the Formula

`InverseTCumulativeDistribution` is calculated using the **Statistics.UtilityFunctions** class. The following table describes this function's parameters and its values.

Method Name	Parameters	Return Value
<code>InverseTCumulativeDistribution</code>	<ol style="list-style-type: none"> <li>1. <b>p</b>: the alpha value (probability).</li> <li>2. <b>degreeOfFreedom</b>: an integer value that represents the degree of freedom.</li> <li>3. <b>oneTail</b>: If true, one-tailed distribution is used; otherwise two-tailed distribution is used.</li> </ol>	A double that represents the Inverse of T cumulative distribution function probability.

#### Example

Here is a code snippet that shows a sample usage.

##### [C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double x= Statistics.UtilityFunctions.
InverseTCumulativeDistribution(p, degreeOfFreedom,OneTail );
```

##### [VB .NET]

```
Imports Syncfusion.Windows.Forms.Chart.Statistics
double x= Statistics.UtilityFunctions.
InverseTCumulativeDistribution(p, degreeOfFreedom,OneTail )
```

### 4.12.2.16 TCumulative Distribution

This formula will returns the T cumulative distribution (student's t-distribution) for a degree of freedom  $> 0$ . When there is a need to estimate the mean of a normally distributed population for a given sample, the t-distribution comes into action. It is the basis of the popular t-tests to find out the difference between two sample means.

For a sample with size  $n$  drawn from a normal population with mean  $\mu$  and standard deviation  $\sigma$ . Let  $\bar{X}$  and  $s$  denote the sample mean and sample standard deviation respectively. Then the quantity

$$Z = \frac{\bar{X}_n - \mu}{\sigma / \sqrt{n}}$$

gives the t-distribution for  $n-1$  degrees of freedom.

### Using the Formula

TCumulativeDistribution is calculated using the **Statistics.UtilityFunctions** class. The following table describes this function's parameters and its values.

Method Name	Parameters	Return Value
TCumulativeDistribution	1. <b>t Value</b> : the T value for which you want the distribution. 2. <b>degreeOfFreedom</b> : an integer value that represents the degree of freedom. 3. <b>oneTail</b> : If true, one-tailed distribution is used; otherwise two-tailed distribution is used.	A double that represents the T cumulative distribution function probability.

### Example

Here is a code snippet that shows a sample usage.

[C#]

```
using Syncfusion.Windows.Forms.Chart.Statistics;
double x= Statistics.UtilityFunctions.TCumulativeDistribution(tvalue,
degreeOfFreedom,OneTail );
```

**[VB .NET]**

```
Imports Syncfusion.Windows.Forms.Chart.Statistics  
double x= Statistics.UtilityFunctions.TCumulativeDistribution(tvalue,  
degreeOfFreedom,OneTail )
```

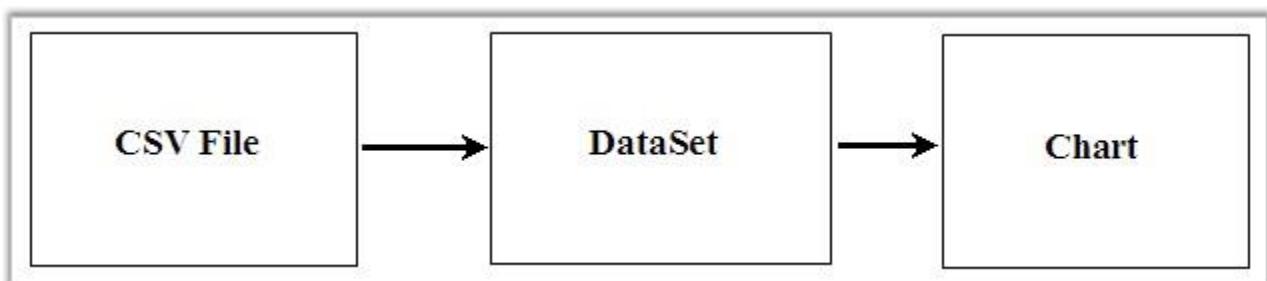
## 4.13 Importing

The Chart provides a simple API to let you populate it with any kind of data, provided, you bring that data from the data store at the runtime. You can then either populate the chart with that data, or bind the chart to the DataTable or DataSet, in which the data is contained. See [ChartData](#) for more information on Data Binding.

In this section, we will illustrate how the data from certain data stores can be brought to the runtime and bound to the chart. In a way this deals more with extracting data from the mentioned data stores than any support in Chart for binding to such data stores.

### 4.13.1 Importing a CSV file

There is no built-in support in Chart for importing data from CSV (Comma Separated Values) files. But this can be easily accomplished by using the "Microsoft.Jet.OLEDB.4.0" to first convert it into a DataSet and then bind it to the chart. This is illustrated in this sample that is distributed with the installation:



*Figure 350: Importing a CSV File*

#### Sample Location:

<..\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Import\Data from CSV>

#### 4.13.2 Import Data from XML to a Chart

There is no built-in support in Chart for importing data from an XML file. But given a corresponding XSLT file, the xml data can be converted into a DataSet, which can then be bound to the chart easily. This is illustrated in this sample that is distributed with the installation:

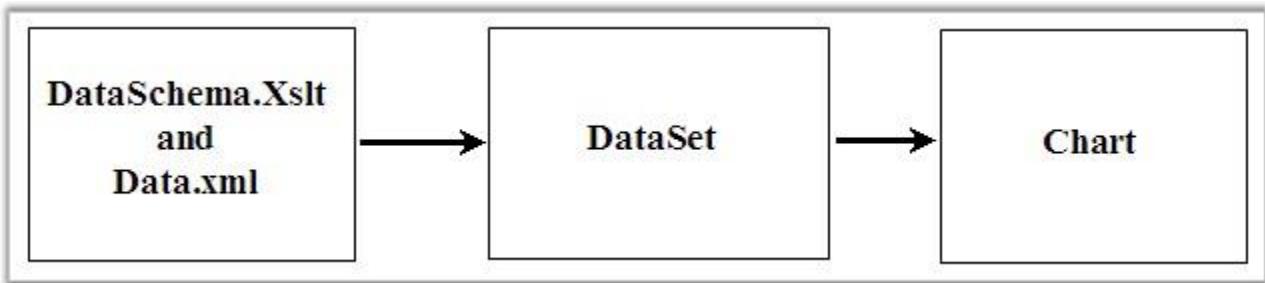


Figure 351: Importing data from XML to Chart

#### Sample Location

[Windows Forms - My Documents\Syncfusion\EssentialStudio\Version NumberWindows\Chart.Windows\Samples\2.0\Import\Import Data from XML](Windows%20Forms%20-%20My%20Documents%20Syncfusion%20EssentialStudio%20Version%20NumberWindows%20Chart.Windows%20Samples%20Import%20Data%20from%20XML)

#### 4.13.3 Import Data from Excel to Chart

There is no built-in support in Chart for importing data from XLS (MS Excel) files. But this can be easily accomplished by using the "Microsoft.Jet.OLEDB.4.0" to first convert it into a DataSet and then bind it to the chart. This is illustrated in this sample that is distributed with the installation:

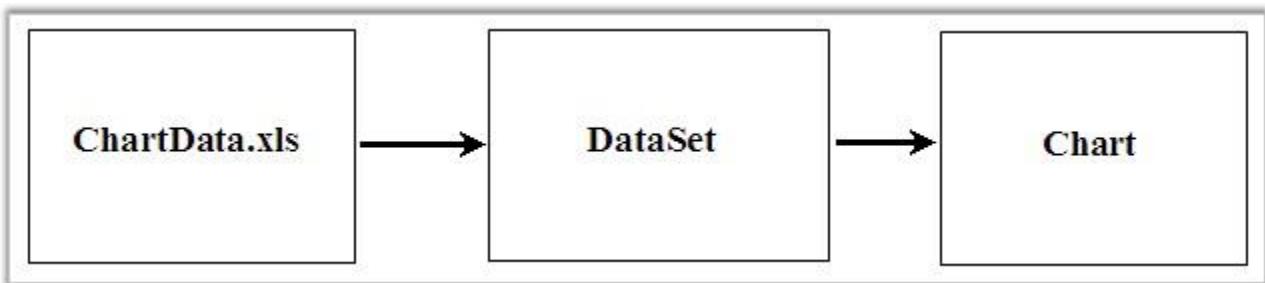


Figure 352: Importing data from Excel to Chart

## **Sample Location**

<My Documents\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Import\Data from Excel>

## **4.14 Exporting**

Essential Chart has built-in support for exporting the chart control into various image formats. Also, using our complementary products like Essential XlsIO, DocIO and PDF you can also export the chart image into Excel, Word Doc and PDF documents.

More on this here:

### **4.14.1 Exporting as an Image**

The chart image can easily be exported as an image file in several different formats.

#### **Programmatically**

##### **[C#]**

```
private string fileName;
fileName = Application.StartupPath + "\\chartexport";
fileName = fileName + ".gif";

this.chartControl1.SaveImage(fileName);

// Launches the file.
System.Diagnostics.Process.Start(fileName);
```

##### **[VB.NET]**

```
Private fileName As String
fileName = Application.StartupPath + "\\chartexport"
fileName = fileName + ".gif"

Me.chartControl1.SaveImage(fileName)
```

```
' Launches the file.  
System.Diagnostics.Process.Start(exportFileName)
```

Based on the filename extension the chart has built-in support to save the image in the following formats.

File Extension	File Type
.bmp	BMP
.jpg	JPEG
.jpeg	JPEG
.gif	GIF
.tiff	TIFF
.Wmf	WMF
.emf	EMF
.svg	SVG (Scalable Vector Graphics)
.eps	Post Script

If the specified extension is none of the above, then the chart is exported as a bitmap.

During runtime, the Chart control can be saved as a file using the **Chart Toolbar** save option.

### **Editable Text Support for EPS Images**

The Chart control can export an EPS image with editable text by setting the **EditableText** property to **true**.

**[C#]**

```
ToPostScript toPostScript = new ToPostScript();  
toPostScript.EditableText = true;  
  
using(Graphics g =  
toPostScript.GetRealGraphics(this.chartControl1.Size))  
{  
    this.chartControl1.Draw(g, this.chartControl1.Size);  
    g.Dispose();
```

```
        toPostScript.Save("EditableTextChart.eps");  
    }
```

**[VB]**

```
Dim toPostScript As New ToPostScript()  
toPostScript.EditableText = True  
  
Using g As Graphics =  
    toPostScript.GetRealGraphics(Me.chartControl1.Size)  
    Me.chartControl1.Draw(g, Me.chartControl1.Size)  
    g.Dispose();  
    toPostScript.Save("EditableTextChart.eps")  
End Using
```

The figure below shows the chart EPS image text editing in Adobe Illustrator.

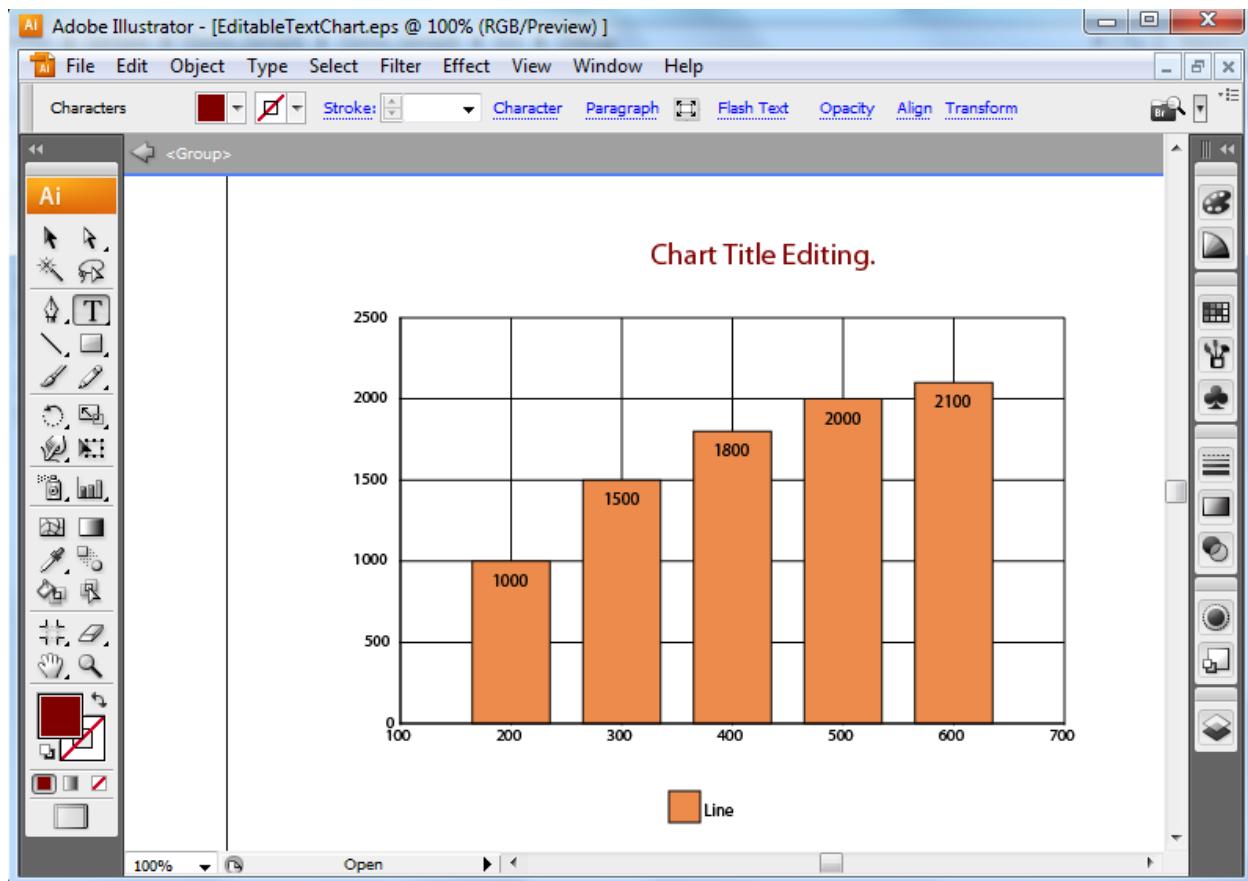


Figure 353: EPS Chart with Editable Text



**Note:** *Chart wrapping and formatting will not be possible in the EPS image by enabling this property.*

#### 4.14.2 Exporting to Word Doc

The chart control can be exported to a Word doc file as an image using Essential DocIO. The chart control provides APIs to convert it to an image, while DocIO lets you insert this image into a Word Document file programmatically.

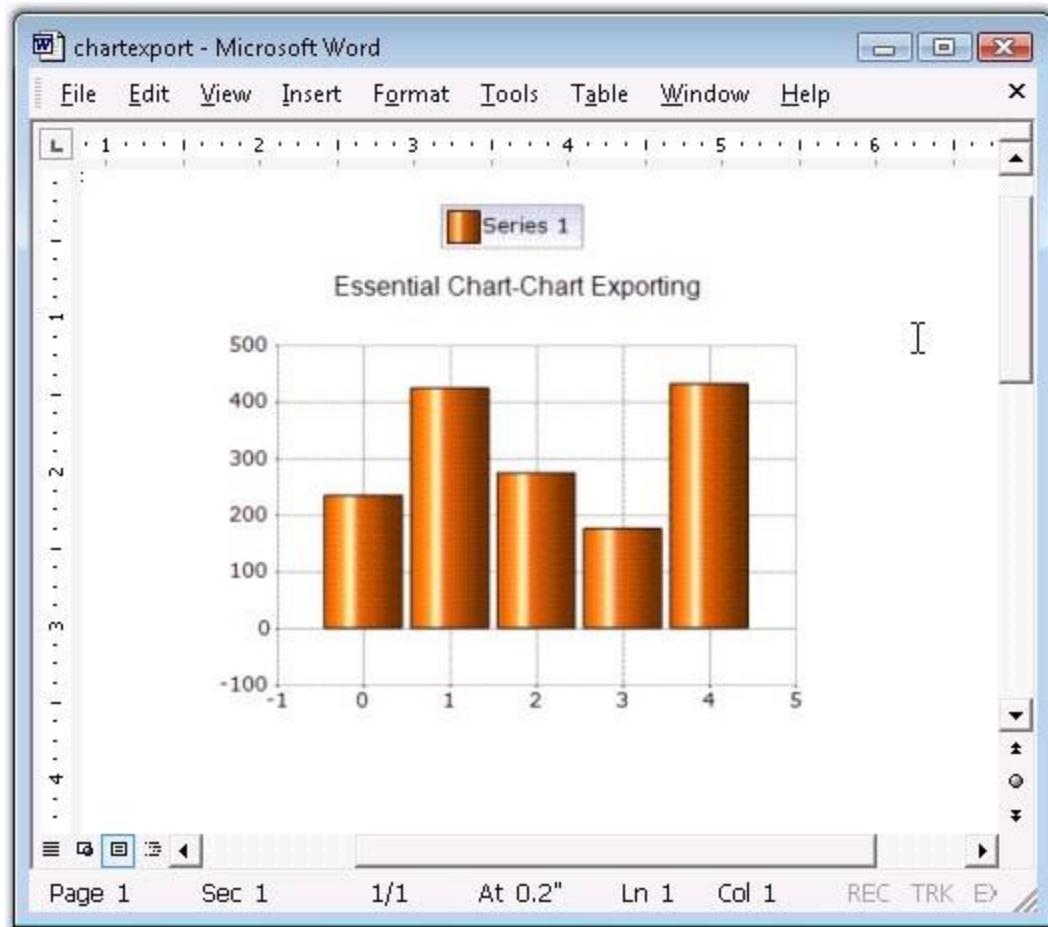


Figure 354: Exporting Chart to DocIO

Given below are the steps that will guide you through this process.

1. Add the Syncfusion.DocIO.Base and Syncfusion.DocIO.Windows assemblies.
2. Add the namespace Syncfusion.DocIO and Syncfusion.DocIO.DLS in your form.

**[C#]**

```
using Syncfusion.DocIO;
using Syncfusion.DocIO.DLS;
```

**[VB .NET]**

```
Imports Syncfusion.DocIO
Imports Syncfusion.DocIO.DLS
```

3. Add the code snippet that is given below in your form.

**[C#]**

```
string fileName=Application.StartupPath+"\\chartexport";
string exportFileName = fileName + ".doc";
string file = fileName + ".gif";

this.chartControl1.SaveImage(file);

// Create a new document.
WordDocument document = new WordDocument();

// Adding a new section to the document.
IWSection section = document.AddSection();
// Adding a paragraph to the section.
IWPParagraph paragraph = section.AddParagraph();

// Writing text.
paragraph.AppendText( "Essential Chart" );

// Adding a new paragraph.
paragraph = section.AddParagraph();
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DLS.HorizontalAlignment.Center;

// Inserting chart.
paragraph.AppendPicture( Image.FromFile(file));
// Save the Document to disk.
document.Save(exportFileName , Syncfusion.DocIO.FormatType.Doc );

// Launches the file.
System.Diagnostics.Process.Start(exportFileName);
```

**[VB.NET]**

```
Dim fileName As String =Application.StartupPath & "\\chartexport"
Dim exportFileName As String = fileName & ".doc"
Dim file As String = fileName & ".gif"

Me.chartControl1.SaveImage(file)

' Create a new document.
Dim document As WordDocument = New WordDocument()
' Adding a new section to the document.
```

```
Dim section As IWSection = document.AddSection()
' Adding a paragraph to the section.
Dim paragraph As IWPParagraph = section.AddParagraph()
' Writing text.
paragraph.AppendText("Essential Chart")
' Adding a new paragraph.
paragraph = section.AddParagraph()
paragraph.ParagraphFormat.HorizontalAlignment =
Syncfusion.DLS.HorizontalAlignment.Center

' Inserting chart.
paragraph.AppendPicture(Image.FromFile(file))

' Save the Document to disk.
document.Save(exportFileName, Syncfusion.DocIO.FormatType.Doc)

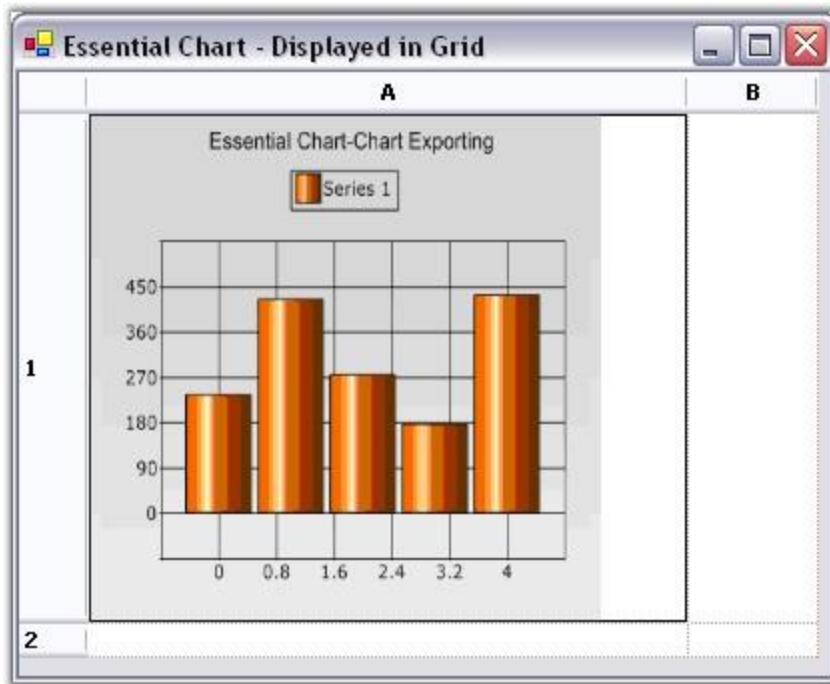
' Launches the file.
System.Diagnostics.Process.Start(exportFileName)
```

A sample demonstrating the above is available in our installation at the following location:

["My Documents\Syncfusion\EssentialStudio\Version  
NumberWindows\Chart.Windows\Samples\2.0\Export\Chart Export Data"](#)

#### **4.14.3 Exporting to Grid**

The chart control can be exported into a grid cell (in Essential Grid) as an image using Essential Grid. The chart control provides APIs to convert it to an image, while the Grid will let you insert this image into any specific cell.



*Figure 355: Exporting Chart to Grid*

The steps that are given below will guide you through the process.

1. Add the Syncfusion.Grid.Base and Syncfusion.Grid.Windows assemblies
2. Add a form (Form2) to hold the Grid control in which the chart is to be exported.
3. Drag a grid control onto the Form2.
4. Add the namespace Syncfusion.Windows.forms.Grid in Form2.

[C#]

```
using Syncfusion.Windows.Forms.Grid;
```

[VB.NET]

```
Imports Syncfusion.Windows.Forms.Grid
```

5. Add the code snippet that is given below in Form2 to get the chart data into the grid.

**[C#]**

```
// Creates a new instance of the ImageList class.  
ImageList img = new ImageList();  
  
// Adds the image to the Image collection of the ImageList.  
img.Images.Add(Image.FromFile(this.Name));  
  
// Specify the size of the image.  
img.ImageSize = new Size(256, 256);  
  
// Set the imagelist of the cell.  
this.gridControl1[1,1].ImageList = img;  
  
// Specify the index for the image to be displayed.  
this.gridControl1[1, 1].ImageIndex = 0;  
  
// Specify the row and column height of the cell.  
this.gridControl1.RowHeights[1] = 300;  
this.gridControl1.ColWidths[1] = 300;  
  
// Specify the image size mode.  
this.gridControl1[1, 1].ImageSizeMode = GridImageSizeMode.CenterImage;
```

**[VB.NET]**

```
' Creates a new instance of the ImageList class.  
Dim img As ImageList = New ImageList()  
  
' Adds the image to the Image collection of the ImageList.  
img.Images.Add(Image.FromFile(Me.Name))  
  
' Specify the size of the image.  
img.ImageSize = New Size(256, 256)  
  
' Set the imagelist of the cell.  
Me.gridControl1(1,1).ImageList = img  
  
' Specify the index for the image to be displayed.  
Me.gridControl1(1, 1).ImageIndex = 0  
  
' Specify the row and column height of the cell.  
Me.gridControl1.RowHeights(1) = 300  
Me.gridControl1.ColWidths(1) = 300
```

```
' Specify the image size mode.  
Me.gridControl1(1, 1).ImageSizeMode = GridImageSizeMode.CenterImage
```

6. Add the code that is given below in the form with the chart control to be exported.

**[C#]**

```
private Form2 gridForm;  
this.gridForm= new Form2();  
  
string fileName=Application.StartupPath+"\\chartexport";  
string file = fileName + ".gif";  
  
if(!System.IO.File.Exists(file))  
this.chartControl1.SaveImage(file);  
  
// Specify the filename as the name of the form.  
gridForm.Name = file;  
  
// Shows the form with grid control with the chart exported.  
gridForm.ShowDialog();
```

**[VB.NET]**

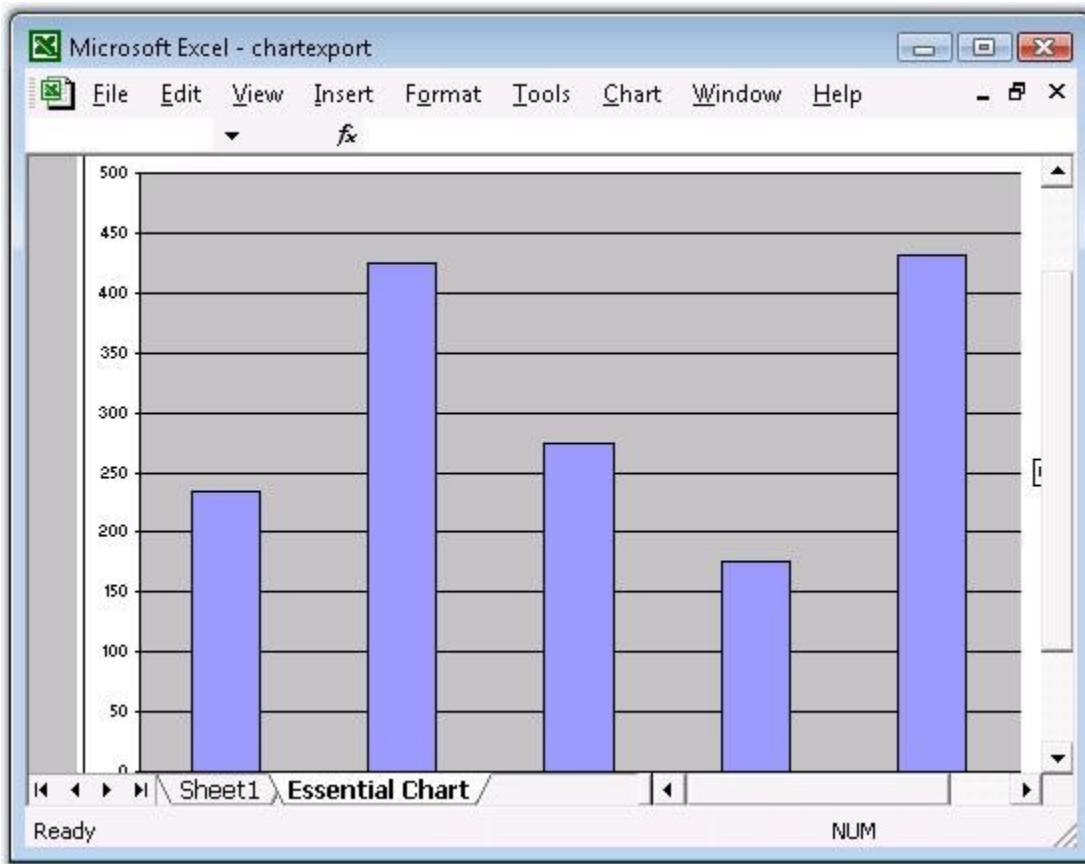
```
Private gridForm As Form2  
Me.gridForm= New Form2()  
  
Dim fileName As String =Application.StartupPath & "\chartexport"  
Dim file As String = fileName & ".gif"  
  
If (Not System.IO.File.Exists(file)) Then  
    Me.chartControl1.SaveImage(file)  
End If  
  
' Specify the filename as the name of the form.  
gridForm.Name = file  
  
' Shows the form with grid control with the chart exported.  
gridForm.ShowDialog()
```

A sample demonstrating the above is available in our installation at the following location:

["My Documents\Syncfusion\EssentialStudio\Version  
NumberWindows\Chart.Windows\Samples\2.0\Export\Chart Export Data"](#)

#### 4.14.4 Exporting to Excel

Essential Chart data can be exported into an Excel document and an Excel chart can be created to use the above data using **Essential XlsIO**. Though there is no built-in support for this, this can be easily implemented with a very intuitive XlsIO API.



*Figure 356: Essential Chart translated into an Excel Chart by using Essential XlsIO*

Given below are the steps that will guide you through this process.

1. Add the Syncfusion.XLsIO.Base and Syncfusion.XLsIO.Windows assemblies.
2. Add the namespace Syncfusion.XLsIO in your form.

```
[C#]  
  
using Syncfusion.XlsIO;
```

**[VB.NET]**

```
Imports Syncfusion.XlsIO
```

3. Add the code snippet that is given below in your form.

**[C#]**

```
string exportFileName = Application.StartupPath+"\charlexport" +  
".xls";  
  
// A new workbook with a worksheet should be created.  
IWorkbook chartBook = ExcelUtils.CreateWorkbook(1);  
IWorksheet sheet = chartBook.Worksheets[0];  
  
// Fill the worksheet with chart data.  
for(int i=1;i<=5;i++)  
{  
    sheet.Range[i,1].Number = this.chartControl1.Series[0].Points[i-  
    1].X;  
    sheet.Range[i,2].Number = this.chartControl1.Series[0].Points[i-  
    1].YValues[0];  
}  
  
// Create a chart worksheet.  
IChart chart = chartBook.Charts.Add("Essential Chart");  
    // Specify the title of the Chart.  
chart.ChartTitle = "Essential Chart";  
  
// Initialize a new series instance and add it to the series collection  
// of the chart.  
IChartSerie series = chart.Series.Add();  
  
// Specify the chart type of the series.  
series.SerieType = ExcelChartType.Column_Clustered;  
  
// Specify the name of the series. This will be displayed as the text  
// of the legend.  
series.Name = "Sample Series";  
  
// Specify the value ranges for the series.  
series.Values = sheet.Range["B1:B5"];  
  
// Specify the Category labels for the series.  
series.CategoryLabels = sheet.Range["A1:A5"];
```

```
// Make the chart as active sheet.  
chart.Activate();  
  
// Save the Chart book.  
chartBook.SaveAs(exportFileName);  
  
chartBook.Close();  
ExcelUtils.Close();  
  
// Launches the file.  
System.Diagnostics.Process.Start(exportFileName);
```

**[VB.NET]**

```
Dim exportFileName As String = Application.StartupPath & "\chartexport"  
& ".xls"  
  
' A new workbook with a worksheet should be created.  
Dim chartBook As IWorkbook = ExcelUtils.CreateWorkbook(1)  
Dim sheet As IWorksheet = chartBook.Worksheets(0)  
  
' Fill the worksheet with chart data.  
For i As Integer = 1 To 5  
    sheet.Range(i,1).Number = Me.chartControl1.Series(0).Points(i-1).X  
    sheet.Range(i,2).Number = Me.chartControl1.Series(0).Points(i-  
    1).YValues(0)  
Next i  
  
' Create a chart worksheet.  
Dim chart As IChart = chartBook.Charts.Add("Essential Chart")  
' Specify the title of the Chart.  
chart.ChartTitle = "Essential Chart"  
  
' Initialize a new series instance and add it to the series collection  
of the chart.  
Dim series As IChartSerie = chart.Series.Add()  
  
' Specify the chart type of the series.  
series.SerieType = ExcelChartType.Column_Clustered  
  
' Specify the name of the series. This will be displayed as the text  
of the legend.  
series.Name = "Sample Series"  
                    ' Specify the value ranges for the series.  
series.Values = sheet.Range("B1:B5")  
' Specify the Category labels for the series.
```

```
series.CategoryLabels = sheet.Range("A1:A5")

' Make the chart as active sheet.
chart.Activate()

' Save the Chart book.
chartBook.SaveAs(exportFileName)
chartBook.Close()
ExcelUtils.Close()

' Launches the file.
System.Diagnostics.Process.Start(exportFileName)
```

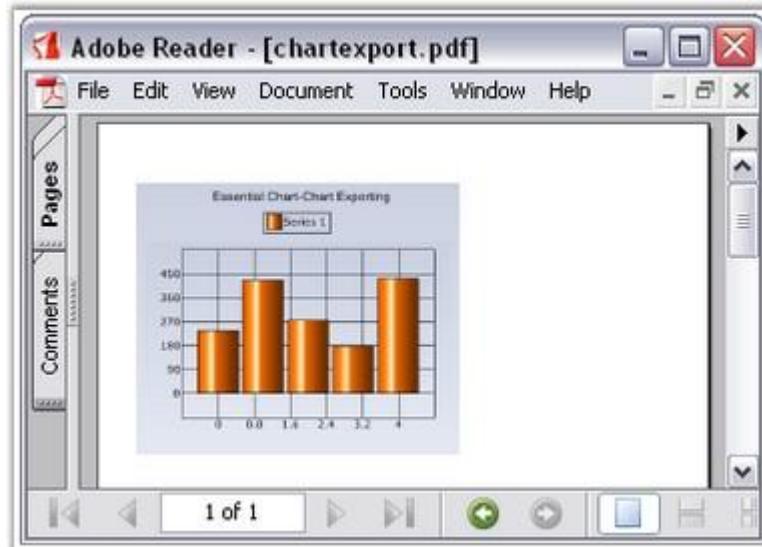
## **Sample**

A sample demonstrating the above functionality is available in our installation at the following location:

["My Documents\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\Export\Chart Export Data"](#)

### **4.14.5 Exporting to PDF**

The chart control can be exported into a PDF file as an image using Essential PDF. The chart control provides APIs to convert it to an image, while Essential PDF lets you insert this image into a Word Document file programmatically.



*Figure 357: Exporting Chart to PDF*

1. 1. Add the Syncfusion.Pdf.Base and Syncfusion.Pdf.Windows assemblies.

2. 2. Add the namespace **Syncfusion.Pdf** in your form.

**[C#]**

```
using Syncfusion.Pdf;
using Syncfusion.Pdf.Graphics;
```

**[VB .NET]**

```
Imports Syncfusion.Pdf
Imports Syncfusion.Pdf.Graphics
```

3. 3. Add the code snippet that is given below in your form.

**[C#]**

```
string fileName=Application.StartupPath+"\\"+chartexport";
string exportFileName = fileName + ".pdf";
string file = fileName + ".gif";
this.chartControl1.SaveImage(file);

//Create a PDF document
PdfDocument pdfDoc = new PdfDocument();

//Add a page to the empty PDF document
pdfDoc.Pages.Add();

//Draw chart image in the first page
pdfDoc.Pages[0].Graphics.DrawImage(PdfImage.FromFile(file), new
PointF(10, 30));

//Save the PDF Document to disk.
pdfDoc.Save(exportFileName);

// Launches the file.
System.Diagnostics.Process.Start(exportFileName);
```

**[VB .NET]**

```
Dim fileName As String = Application.StartupPath & "\chartexport"
```

```
Dim exportFileName As String = fileName & ".pdf"
Dim file As String = fileName & ".gif"
Me.chartControl1.SaveImage(file)

'Create a PDF document
Dim pdfDoc As PdfDocument = New PdfDocument()

'Add a page to the empty PDF document
pdfDoc.Pages.Add()

'Draw chart image in the first page
pdfDoc.Pages(0).Graphics.DrawImage(PdfImage.FromFile(file), New
PointF(10, 30))

'Save the PDF Document to disk.
pdfDoc.Save(exportFileName)

' Launches the file.
System.Diagnostics.Process.Start(exportFileName)
```

A sample demonstrating the above is available in our installation at the following location:

["My Documents\Syncfusion\EssentialStudio\Version  
Number\Windows\Chart.Windows\Samples\2.0\Export\Chart Export Data"](#)

## 4.15 Design time Features

The design time features are discussed in the below topics:

### 4.15.1 Chart Templates

Essential Chart is provides support to save the series and point properties as XML file. This enables you to save the series and point properties into chart template and load the chart templates into the Chart control when needed.

#### Use Case Scenarios

When you want to create charts with consistent look and feel, you can utilize this feature. You can load the saved chart source to achieve this.

#### Working with Chart Template

Essential Chart is now associated with the creation and loading of chart templates into the ChartControl. It provides easy methods to save and load the templates. This section will walk you through the saving, loading and resetting of the chart templates and the various benefits of using it.

### Benefits

- Aesthetic items like appearance, positioning etc., of a chart can be saved in the template.
- Appearance settings saved in a Chart Template is reusable.
- Also stores any static data, if available in the chart.
- The user can save the existing structure of the chart control to an .xml file format.
- All the charts in your applications can be created with consistent look and feel.

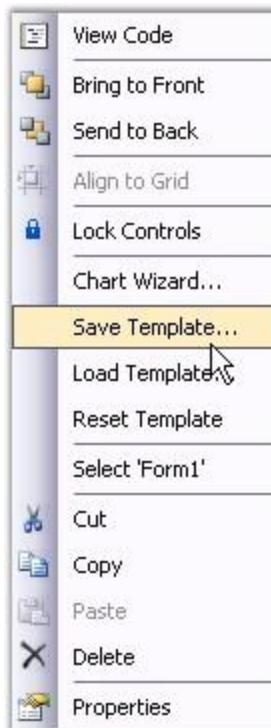


Figure 358: Saving Template through Context Menu

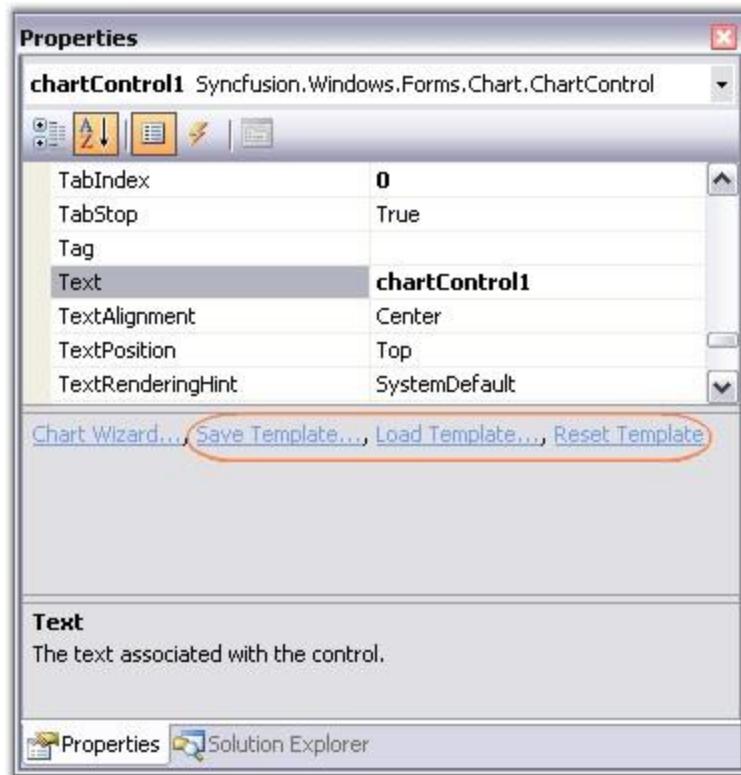


Figure 359: Saving, Loading and Resetting Templates through Designer Verbs

### Save Template

The appearance settings for various components of a Chart like ChartSeries, ChartArea, Series properties and Point properties can be stored in a template, which can be loaded into new Chart control when needed.

A chart template can contain the properties of more than one data series. When such templates are loaded into a destination ChartControl, the appearance settings of the data series will be applied in a sequential order, i.e., the first set of appearance settings of a data series will be applied to the destination Chart's first series and the second set of appearance properties of the data series will be applied to the destination Chart's second series and so on.

If the destination collection's length is larger than the source collection, then the settings will repeat itself for these additional entries in the destination collection.

These Charts can be saved as templates in the below two ways.

- Selecting the **Save Template** option from the context menu as shown above.
- By clicking the **Save Template** designer verb in the Visual Studio property browser as shown above.

- ChartTemplate has a static method to save the data programmatically. We need to pass ChartControl instance and a file name(it can accept stream file also.), through this save method.

**[C#]**

```
ChartTemplate.Save(this.chartControl1, "TemplateName.xml");
```

**[VB]**

```
ChartTemplate.Save(Me.chartControl1, "TemplateName.xml")
```

### Load Template

Essential Chart provides support to load the saved Chart template into a new chart control. This loads the eries properties and the point properties, which was saved in a XML file and applies these properties into the new chart control.

- At the design time, by selecting the **Load Template** from the context menu.
- By clicking the **Load Template** designer verb, in the Visual Studio property browser.

ChartTemplate has static method, to load the template data programmatically. We need to pass the ChartControl, that will be applied with the loaded template data.

### Reset Template

The ChartControl, which when loaded with a template will be applied with the appearance and other settings that were stored in the template. These settings can be reset and the Chart can be reverted back to its original appearance by using the below two methods.

- At the design time, by selecting the "Reset Template.." from the context menu.
- By clicking the "Reset Template" link in the Visual Studio property browser.

ChartTemplate can be reset using the following simple statements,

**[C#]**

```
ChartTemplate ct = new ChartTemplate();
ct.Reset(this.chartControl1);
```

**[VB]**

```
ChartTemplate ct = New ChartTemplate()
```

```
ct.Reset(Me.chartControl1)
```

#### **Sample Link**

To view a sample:

1. Open the **Syncfusion Dashboard**.
2. Select **User Interface > Windows Forms**.
3. Click **Run Samples**.
4. Navigate to **Chart samples > User Interaction > ChartSerialization**.

## **4.15.2 Tasks Window**

The tasks window has sufficient properties exposed in the right manner for users to be able to get started intuitively.

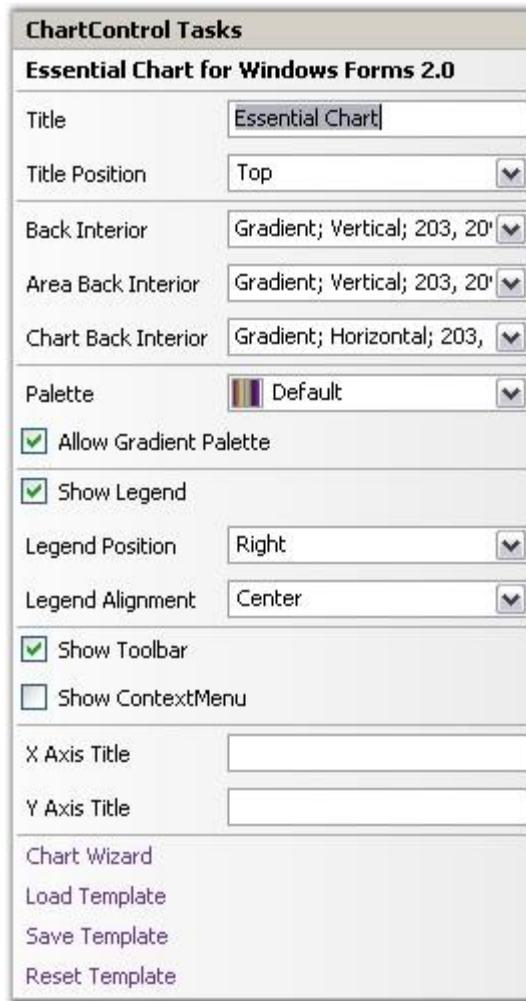


Figure 360: ChartControl Tasks Window

## Tasks

### Title

Used to add title to the Chart directly from the Tasks Window.

### Title Position

Specifies the position relative to the chart at which to render the chart title panel.

ChartTitle.Position Property	Description
------------------------------	-------------

Top	Above the chart; default setting.
Left	Left of the chart.
Right	Left of the chart.
Bottom	Left of the chart.
Floating	Will not be docked to any specific location. Can be docked manually by dragging the title panel.

**Back Interior**

Specifies background brush of the control.

**Area Back Interior**

Specifies background brush of Chart Area of the control.

**Chart Back Interior**

Specifies background brush of ChartInterior.

**Palette**

The Palette that is to be used to provide default colors for the chart series and other chart elements. **Allow Gradient Palette** property is used to enable or disable the gradient values of the palettes.

**ShowLegend**

Specifies if the legend is to be displayed or not.

**Legend Position**

Configuration information of the Legend object.

ChartLegend.Position Property	Description
<b>Top</b>	Positions the legend panel to the Top of Chart.
<b>Left</b>	Positions the legend panel to the Left of Chart.

<b>Right</b>	Positions the legend panel to the Right of Chart.
<b>Bottom</b>	Positions the legend panel to the Bottom of Chart.
<b>Floating</b>	Will not be docked to any specific location. Can be docked manually by dragging the Legend panel.

### **Legend Alignment**

Gets or sets the legend alignment.

<b>ChartLegend.Alignment Property</b>	<b>Description</b>
<b>Center</b>	Aligns to center of chart.
<b>Far</b>	Aligns to Far of chart.
<b>Near</b>	Aligns to Near of chart.

### **Show ToolBar**

Specifies if the ToolBar is to be displayed or not.

### **Show ContextMenu**

Specifies if the ToolBar context menu is to be displayed or not.

### **X Axis Title**

Specifies the Title of Primary x axis.

### **Y Axis Title**

Specifies the Title of Primary y axis.

### **Spacing Between Points**

Specifies the [spacing](#) between the series points.

## 4.16 Printing and Print Preview

### Print Preview

The chart provides a **PrintDocument** that can be sent to the .NET **PrintPreviewDialog** to get a preview of the chart that gets printed. Here is some code that shows how this is done.

[C#]

```
PrintPreviewDialog printPreviewDialog1 = new PrintPreviewDialog();
printPreviewDialog1.Document = this.chartControl1.PrintDocument;
printPreviewDialog1.ShowDialog();
```

[VB .NET]

```
Me.printPreviewDialog1 = New System.Windows.Forms.PrintPreviewDialog
printPreviewDialog1.Document = Me.chartControl1.PrintDocument
printPreviewDialog1.ShowDialog()
```

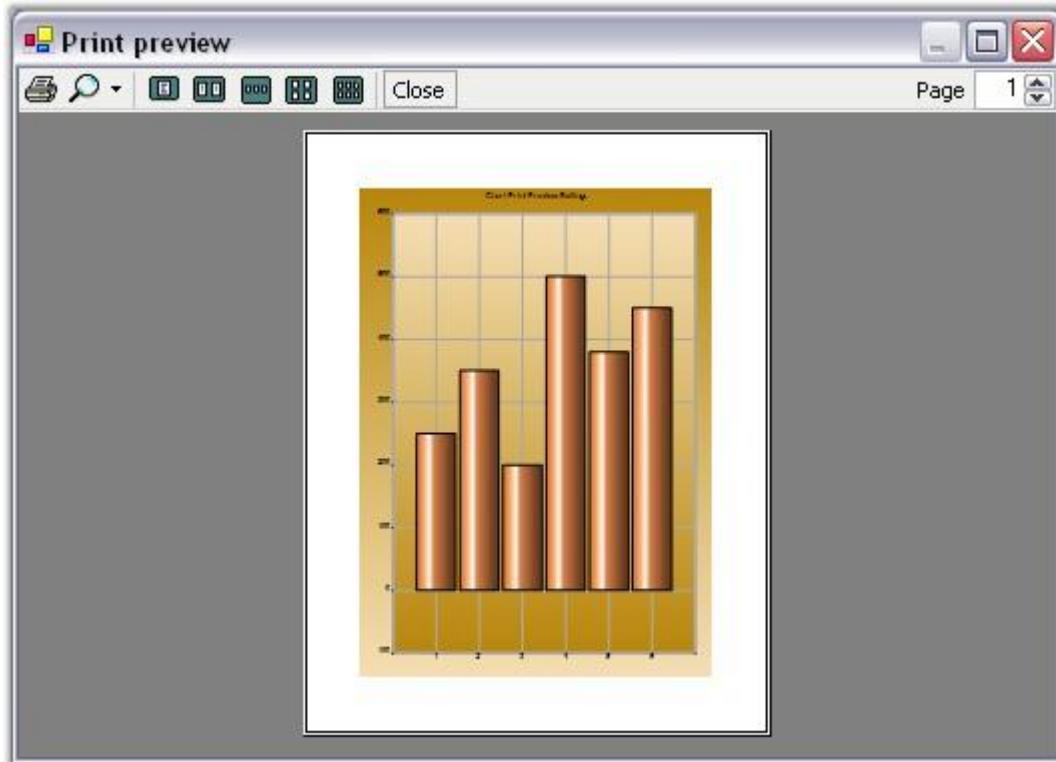


Figure 361: Print Preview Dialog Box

## Printing

Print a chart control using the **PrintDocument** exposed by the chart control as follows:

[C#]

```
this.chartControl1.PrintDocument.Print();
```

[VB .NET]

```
Me.chartControl1.PrintDocument.Print()
```

You can also specify if you want to print the chart in Color or GrayScale using this property.

Chart control Property	Description
PrintColorMode	Indicates the color mode during printing. Possible Values: <ul style="list-style-type: none"><li>• <b>Color</b> - Always Print in Color.</li><li>• <b>GrayScale</b> - Always Print using GrayScale.</li><li>• <b>CheckPrinter</b> - If printer allows color print in color, otherwise use grayscale (default setting).</li></ul>

[C#]

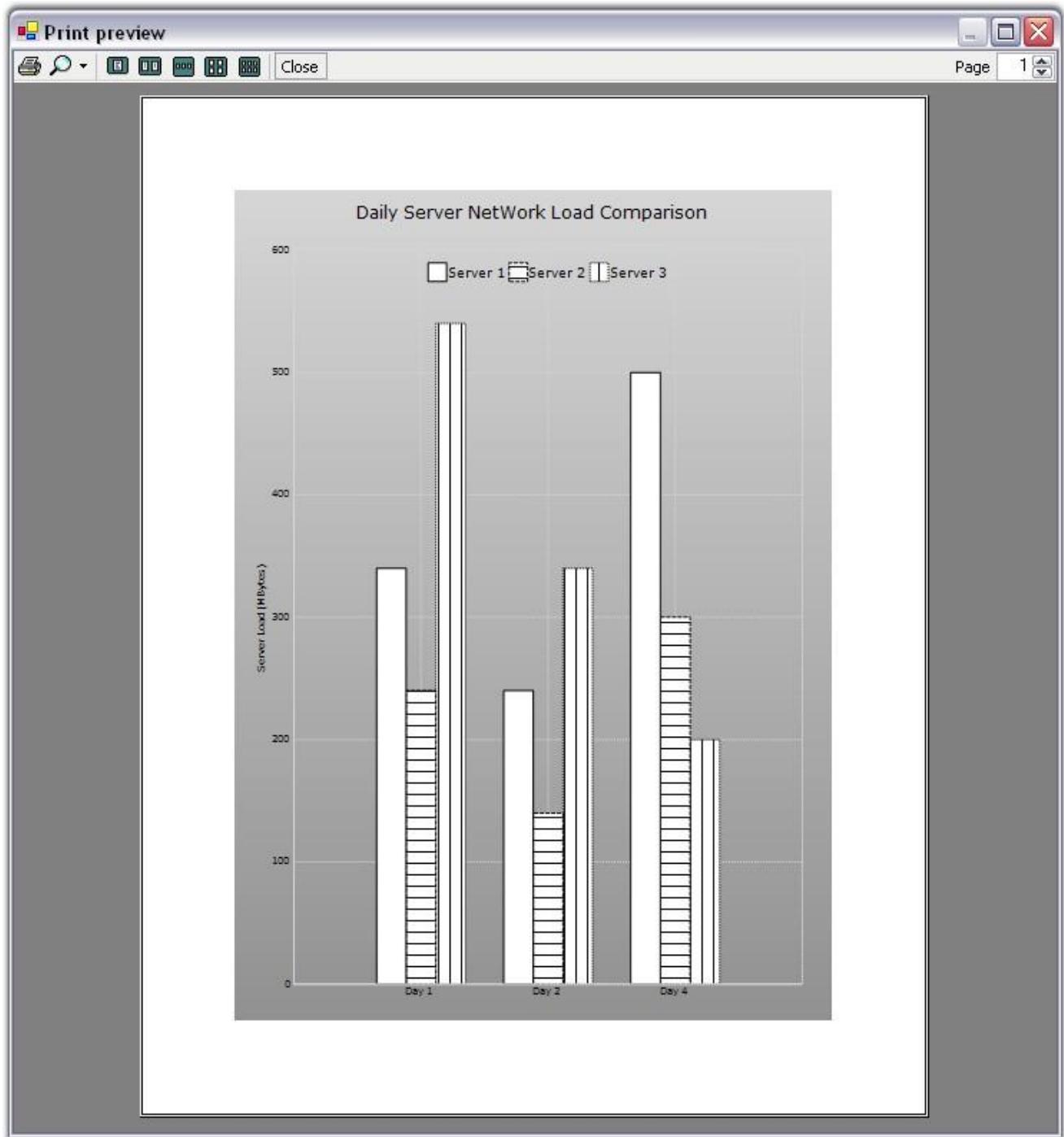
```
this.chartControl1.PrintColorMode = ChartPrintColorMode.GrayScale;
```

[VB .NET]

```
Me.chartControl1.PrintColorMode = ChartPrintColorMode.GrayScale
```

## Automatic Grayscale Handling

Setting **GrayScale** print mode for the chart, lets you print the chart in a gray scale and when multiple series are printed in this case, chart data points are automatically rendered with a patterned brush to differentiate the different series as shown in the image below.



*Figure 362: Column Chart with 2nd and 3rd Series rendered with Patterned Brush*

A sample illustrating the printing features is available in the below location.

..\My Documents\Syncfusion\EssentialStudio\Version  
Number\Windows\Chart.Windows\Samples\2.0\Print\Chart Print

### Displaying ToolBar while printing

**ShowToolBar** property should be set to **true** to display a toolbar in the Chart. You can show or hide the toolbar while printing a Chart using **PrintToolBar** property.

#### [C#]

```
chartControl1.ShowToolbar = true;  
chartControl1.PrintDocument.PrintToolBar = true;
```

#### [VB .NET]

```
chartControl1.ShowToolbar = True  
chartControl1.PrintDocument.PrintToolBar = True
```

## 4.17 Data Manipulation

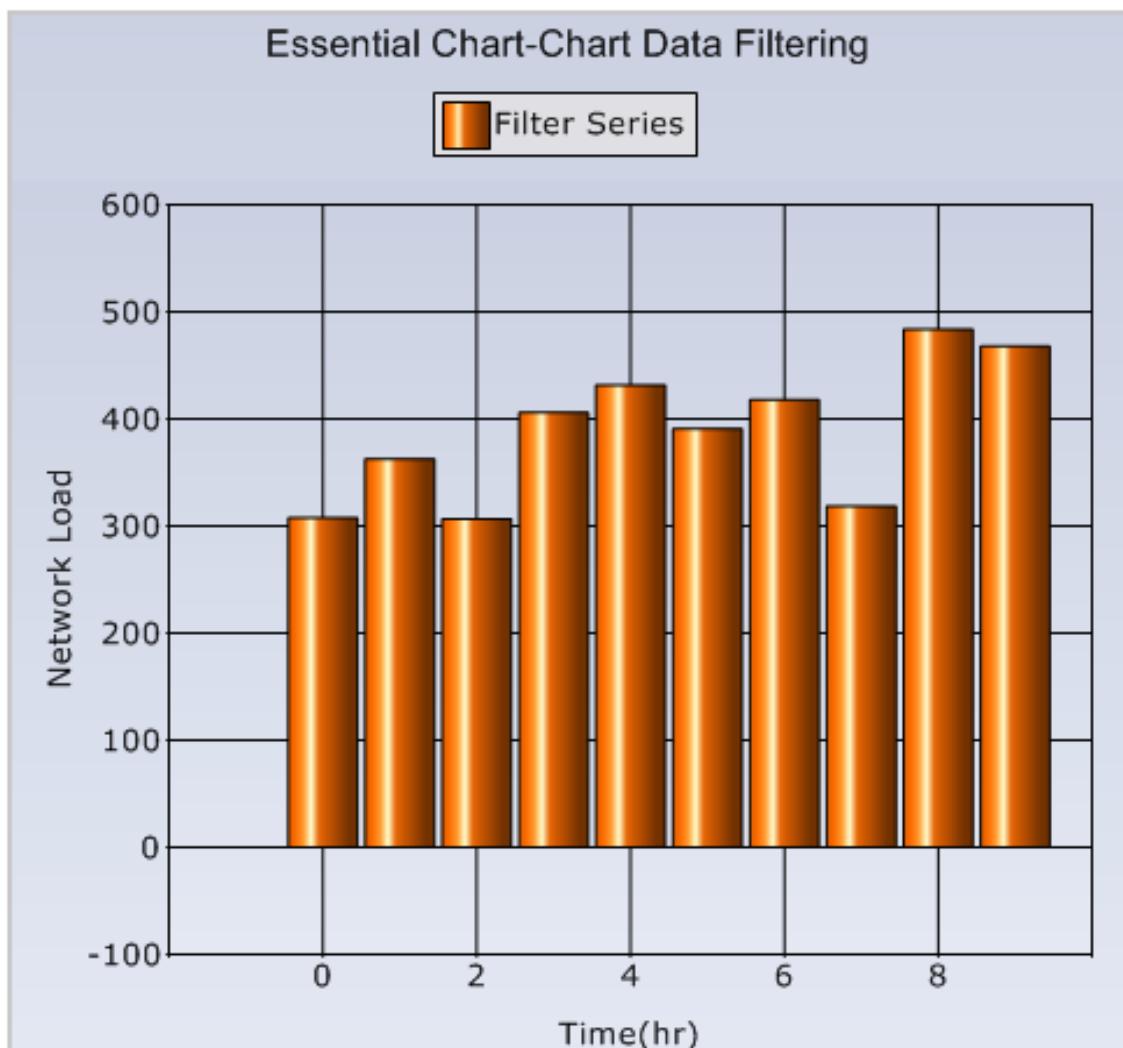
Essential Chart provides a series model implementation that works directly on top of grouped data. Filters, summaries and computed expressions are all supported in Essential Chart and users can easily add custom summaries and filters and have such data displayed in the chart.

### Grouping Support

The enterprise version of Essential Chart includes Essential Grouping that allows Essential Chart to implement a series model that works directly on top of grouped data. All the key advantages of Essential Grouping carry over into the grouping support in Essential Chart. Filters, summaries and computed expressions are all supported in Essential Chart. With Essential Chart, you are not restricted to predefined summaries or filters. You can easily add custom summaries and filters and have such data displayed in the chart.

The following image displays stock data that is grouped by symbol to calculate the total volume. The data contains discrete transaction details with symbol information, volume, and price.

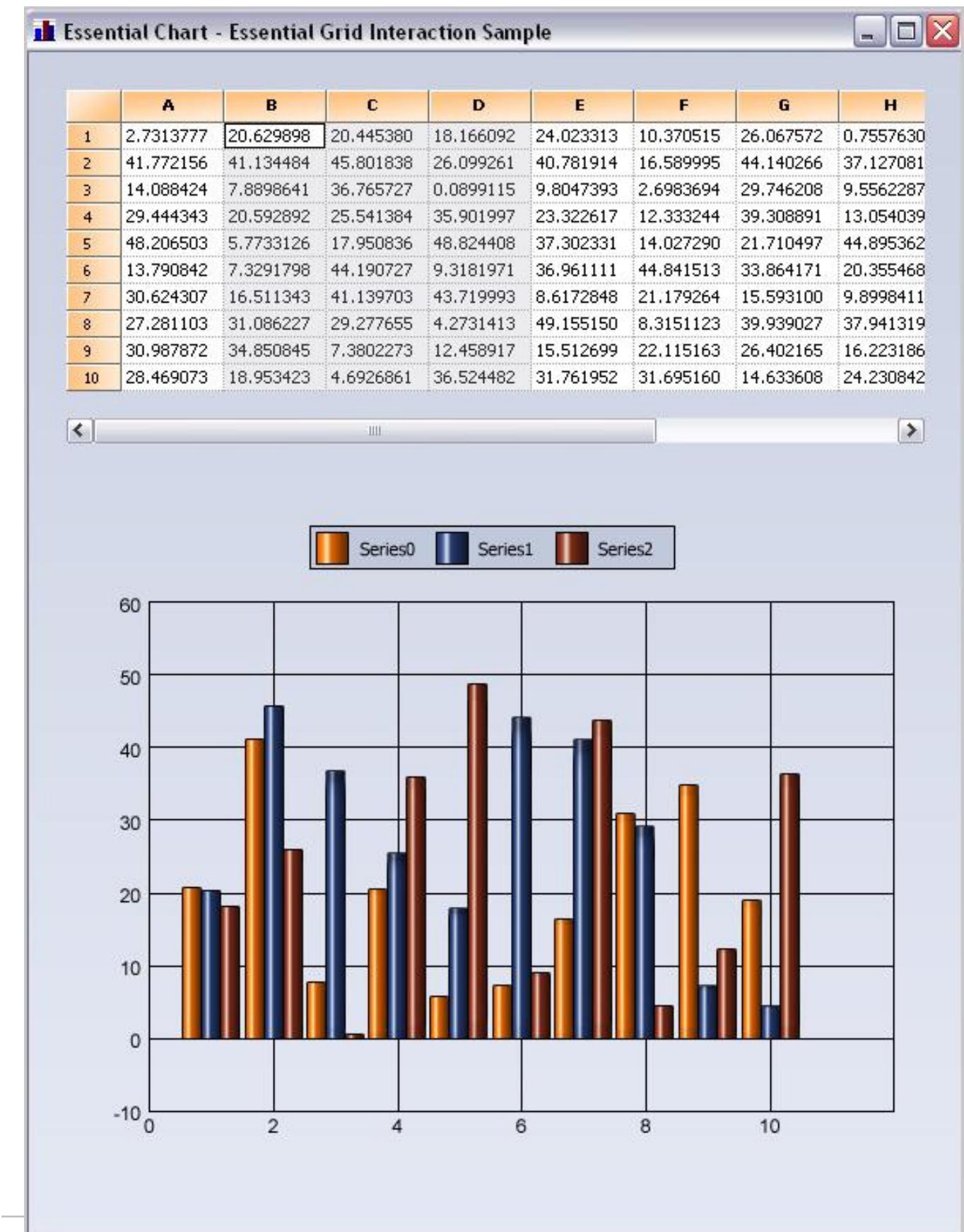
The following image displays the same data with data from transactions that occurred in the morning filtered out. Filtering is done entirely in the grouping engine and the data changes are reflected in the chart. You will never be looking at data that is a filtered and grouped copy. Instead, you are always working with live data with the grouping layer provided by the grouping engine. Any changes made in the underlying data will be immediately reflected in the chart.



*Figure 363: Grouped Data with Filtering*

### **Essential Grid Interaction**

Essential Chart offers great interaction capabilities with Essential Grid. You can use a common data model for the grid and chart. The grid can also serve as a data model for the chart, as shown in the following image. Selected columns are automatically mapped into the chart. All it takes, is a few lines of code to implement a model that adapts the data in question (in this case grid cells) for display in the chart.



*Figure 364: Essential Chart with Essential Grid*

## 4.18 Hit Testing

The section covers the below topics:

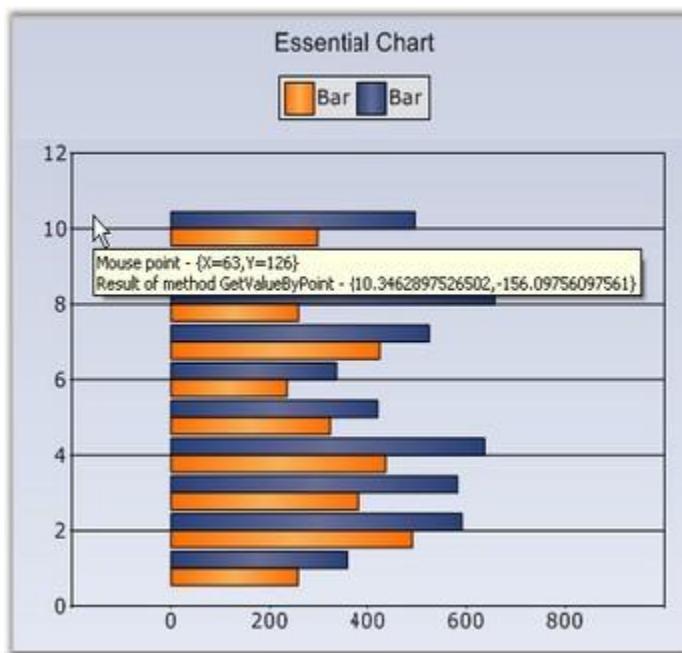
### 4.18.1 Chart Coordinates by Point

#### Chart Coordinates by point

##### GetValueByPoint()

Using the **GetValueByPoint** method, the mouse position in chart client-coordinates can be converted into a corresponding Chart Coordinate in terms of x, y values.

The below figure shows a chart where the tooltip text for each point shows the corresponding x, y value at that position.



*Figure 365: Chart displaying Coordinate value at a Client Point*

**Code snippet for the above sample**

[C#]

```
// Chartcontrol mouse move event.  
private void chartControl_MouseMove(object  
sender, System.Windows.Forms.MouseEventArgs e)  
{  
    ChartPoint chpt = this.chartControl.ChartArea.GetValueByPoint(  
new Point( e.X, e.Y ) );  
    string text = "Result of method GetValueByPoint - {" +  
chpt.ToString() + "," + chpt.YValues[0].ToString() + "}" ;  
    toolTip.SetToolTip( chartControl, text );  
}
```

[VB .NET]

```
' ChartControl mouse move event.  
Private Sub chartControl_MouseMove(ByVal sender As Object, ByVal e As  
System.Windows.Forms.MouseEventArgs)  
    Dim chpt As ChartPoint =  
Me.chartControl.ChartArea.GetValueByPoint(New Point(e.X, e.Y))  
    Dim [text] As String = "Result of method GetValueByPoint - {" +  
chpt.ToString() + "," + chpt.YValues(0).ToString() + "}"  
    toolTip.SetToolTip(chartControl, text)  
End Sub
```

**GetPointByValue()**

The **GetPointByValue** method does the opposite of the above - given a chart coordinate it returns the client co-ordinate corresponding to that chart point.

## 4.18.2 LegendItem By Point

### Get LegendItem By Point

The **Legend.GetItemBy** method will let you get the reference to a legend item at a specific point. Implementing the below code sample, will display a tooltip with legend item name, on which the user mouse hover.

[C#]

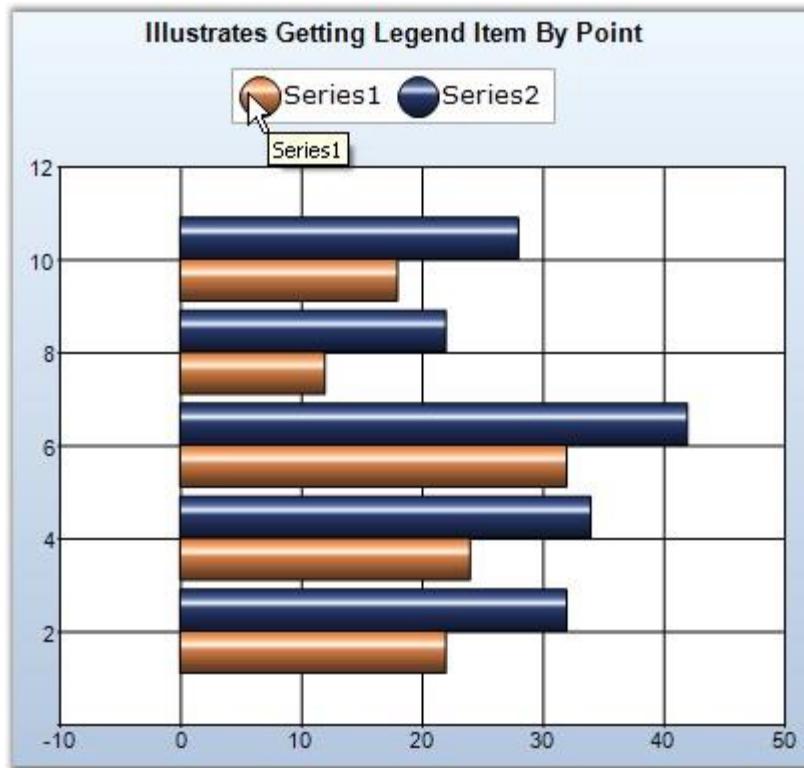
```
private ToolTip toolTip2;
this.chartControl1.Legend.MouseHover += new
MouseEventHandler(lgnd_MouseHover);

void lgnd_MouseHover(object sender, EventArgs e)
{
    Point p1 = this.chartControl1.Legend.PointToClient(new
Point(Control.MousePosition.X, Control.MousePosition.Y));
    ChartLegendItem item = chartControl1.Legend.GetItemBy(p1);
    if (item != null)
        this.toolTip2.Show(item.Text, this.chartControl1.Legend, p1.X +
10, p1.Y + 20, 3000);
}
```

**[VB.NET]**

```
private toolTip2 As ToolTip
AddHandler Me.chartControl1.Legend.MouseHover, AddressOf
lgnd_MouseHover

Private Sub lgnd_MouseHover(ByVal sender As Object, ByVal e As
EventArgs)
    ' Get the item at the specified location..
    Dim p1 As Point = Me.chartControl1.Legend.PointToClient(New
Point(Control.MousePosition.X, Control.MousePosition.Y))
    Dim item As ChartLegendItem = chartControl1.Legend.GetItemBy(p1)
    If item IsNot Nothing Then
        Me.toolTip2.Show(item.Text, this.chartControl1.Legend, p1.X +
10, p1.Y + 20, 3000)
    End If
End Sub
```



*Figure 366: Legend Item Identified using GetItemBy Method*

### 4.18.3 Chart Area Bounds

#### Full Chart Area Bounds

Use the **Bounds** property to get the rectangular area comprising the chart area that includes the axis, axis titles and other sections.

[C#]

```
this.chartControl1.ChartAreaPaint += new  
System.Windows.Forms.PaintEventHandler(chartControl1_ChartAreaPaint);  
  
void chartControl1_ChartAreaPaint(object sender,  
System.Windows.Forms.PaintEventArgs e)  
{  
    Rectangle axisBounds = this.chartControl1.ChartArea.Bounds;  
    // Render a rectangle around this bounds  
    e.Graphics.DrawRectangle(Pens.Red, axisBounds);  
}
```

**[VB.NET]**

```
AddHandler Me.chartControl1.ChartAreaPaint, AddressOf  
chartControl1_ChartAreaPaint  
  
Private Sub chartControl1_ChartAreaPaint(ByVal sender As Object, ByVal  
e As System.Windows.Forms.PaintEventArgs)  
    Dim axisBounds As Rectangle = Me.chartControl1.ChartArea.Bounds  
    ' Render a rectangle around this bounds  
    e.Graphics.DrawRectangle(Pens.Red, axisBounds)  
End Sub
```

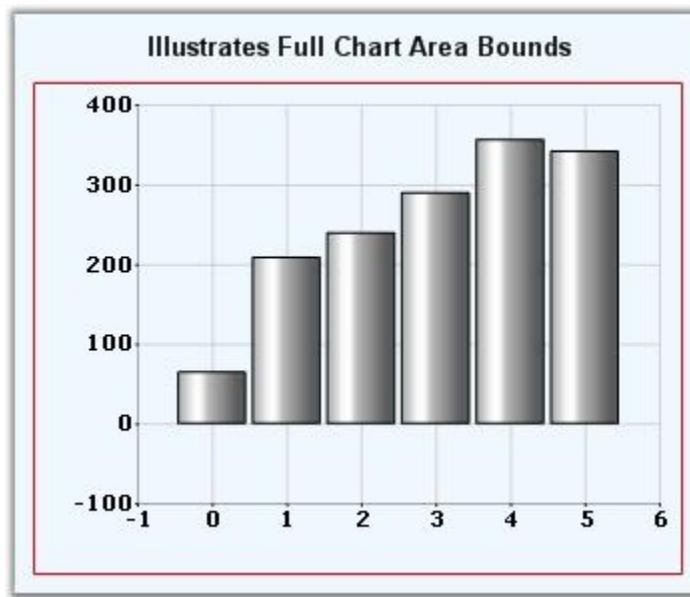


Figure 367: Chart Area Bounds in Red

### Chart Plot Area Bounds

Use the **RenderBounds** property to get the rectangular area comprising just the plot-area, bound by the axes.

**[C#]**

```
this.chartControl1.ChartAreaPaint += new  
System.Windows.Forms.PaintEventHandler(chartControl1_ChartAreaPaint);  
  
void chartControl1_ChartAreaPaint(object sender,  
System.Windows.Forms.PaintEventArgs e)  
{
```

```
Rectangle axisBounds = this.chartControl1.ChartArea.RenderBounds;
// Render a rectangle around this bounds
e.Graphics.DrawRectangle(Pens.Red, axisBounds);
}
```

**[VB.NET]**

```
AddHandler Me.chartControl1.ChartAreaPaint, AddressOf
chartControl1_ChartAreaPaint

Private Sub chartControl1_ChartAreaPaint(ByVal sender As Object, ByVal
e As System.Windows.Forms.PaintEventArgs)
    Dim axisBounds As Rectangle =
Me.chartControl1.ChartArea.RenderBounds
    ' Render a rectangle around this bounds
    e.Graphics.DrawRectangle(Pens.Red, axisBounds)
End Sub
```

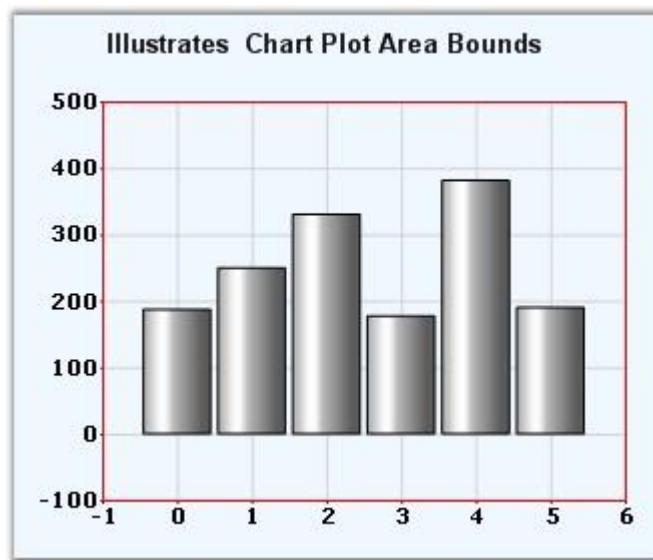


Figure 368: Chart Plot Area Bounds in Red

## 4.19 Chart control events

The following events are discussed in this section:

## 4.19.1 Chart Region Events

The Chart handles the following mouse related events when the user interacts with the Chart using mouse, on certain specific regions in the Chart - Axis Labels, Chart Points or a custom region.

- ChartRegionClick Event
- ChartRegionMouseEnter Event
- ChartRegionMouseHover Event
- ChartRegionMouseMove Event
- ChartRegionMouseLeave Event
- ChartRegionMouseUp Event
- ChartRegionMouseDown Event
- ChartRegionDoubleClick Event

The above events are raised with a **ChartRegionMouseEventArgs** that contain the following properties.

ChartRegionMouseEventArgs Property	Description
Point	Represents the client point where the event occurred.
Region (Expanded below)	Returns the region associated with this event.
Button	Returns the right mouse button actions.

The **Region** property above includes several useful information about the kind of region the user is currently interacting with:

ChartRegion Property	Description
Description	A text description of this region.
Type	Specifies the type of region. Possible values: <ul style="list-style-type: none"> <li>• <b>SeriesPoint</b> - interacted on a data point.</li> <li>• <b>HorAxisLabel</b> - interacted on a horizontal axis</li> <li>• <b>VerAxisLabel</b> - interacted on a vertical axis</li> <li>• <b>ChartCustom</b> - interacted with a region that is none of the above.</li> </ul>
IsChartPoint	Indicates whether the region is a Chart Point in the ChartSeries. This simply checks if the above mentioned Type is SeriesPoint.

SeriesIndex	The index into the <b>Series</b> array of the Chart in which this point occurs. If the Type is SeriesPoint.
PointIndex	The index into the <b>Points</b> array of the ChartSeries in which this point occurs. If the Type is SeriesPoint.
Region	The client region that represents this logical region.
ToolTip	Specifies the tooltip for this region.

**ChartRegionDoubleClick and ChartRegionMouseDown Events**

**[C#]**

```
//ChartRegionDoubleClick Event
this.chartControl1.ChartRegionDoubleClick += new
Syncfusion.Windows.Forms.Chart.ChartRegionMouseEventHandler(this.chartC
ontroll_ChartRegionDoubleClick);

private void chartControl1_ChartRegionDoubleClick(object sender,
ChartRegionMouseEventArgs e)
{
    if (this.chkRegionDoubleClick.Checked)
    {
        if (e.Region.SeriesIndex == 0)
        {
            OutputText(String.Format("Double Click over Series 1 Column
{0} Point : {1}", e.Region.PointIndex, e.Point));
            ShowChartRegion("ChartSeries");
        }
        else
        {
            OutputText(String.Format("Double Click over {0}",
e.Region.Description.ToString()));
            ShowChartRegion(e.Region.Description.ToString());
        }
    }
}

//Usage of Button property in ChartRegionMouseDown Event
void chartControl1_ChartRegionMouseDown(object sender,
ChartRegionMouseEventArgs e)
{
    if(e.Button==MouseButtons.Right)
        Console.WriteLine("Chart Region Mouse Down:=" + e.Point.ToString());
}
```

**[VB .NET]**

```
'ChartRegionDoubleClick Event
AddHandler Me.chartControl1.ChartRegionDoubleClick, AddressOf
Me.chartControl1_ChartRegionDoubleClick

Private Sub chartControl1_ChartRegionDoubleClick(ByVal sender As
Object, ByVal e As ChartRegionMouseEventArgs)
    If Me.chkRegionDoubleClick.Checked Then
        If e.Region.SeriesIndex = 0 Then
```

```
        OutputText([String].Format("Double Click over Series 1
Column {0} Point : {1}", e.Region.PointIndex, e.Point))
        ShowChartRegion("ChartSeries")
    Else
        OutputText([String].Format("Double Click over {0}",
e.Region.Description.ToString()))
        ShowChartRegion(e.Region.Description.ToString())
    End If

End If
End Sub

'Usage of Button property in ChartRegionMouseDown Event
Private Sub chartControl1_ChartRegionMouseDown(ByVal sender As Object,
ByVal e As ChartRegionEventArgs)
    If e.Button = MouseButtons.Right Then
        Console.WriteLine("Chart Region Mouse
Down:=" + e.Point.ToString())
    End If
End Sub
```

## 4.19.2 VisibleRangeChanged Event

ChartControl provides various zooming options for the user while interacting with the Chart. The VisibleRangeChanged event will be raised when the visible range changes during the zooming operation.

### [C#]

```
private static void chartControl1_VisibleRangeChanged(object sender,
EventArgs e)
{
    Console.WriteLine("Visible Range Changed event is raised");
}
```

### [VB .NET]

```
Private Sub chartControl1_VisibleRangeChanged(ByVal sender As Object,
ByVal e As EventArgs)
    Console.WriteLine("Visible Range Changed event is raised")
End Sub
```

### 4.19.3 ChartFormatAxisLabel Event

This event is discussed in detail in this topic: [Customizing Label Text](#).

### 4.19.4 PrepareStyle Event

When a series point is about to be rendered by the chart, it will raise this event and allow event subscribers to change the style used.

[C#]

```
//Listen to the prepare style event for the series.  
series.PrepareStyle += new  
ChartPrepareStyleInfoHandler(series_Preparesyle);  
  
private void series_Preparesyle(object sender,  
ChartPrepareStyleInfoEventArgs args)  
{  
    ChartSeries series = sender as ChartSeries ;  
    if(series != null)  
    {  
        //Condition to select members (data points) who made 100 % quota  
        //in sales  
        if(((series.Points[args.Index].YValues[0] / 150) * 100) >= 100)  
        {  
            args.Style.Interior = new  
Syncfusion.Drawing.BrushInfo(Syncfusion.Drawing.GradientStyle.Horizontal,  
System.Drawing.Color.DarkGreen,  
System.Drawing.Color.LightYellow);  
        }  
    }  
}
```

[VB.NET]

```
'Listen to the prepare style event for the series.  
AddHandler series.PrepareStyle, AddressOf  
Me.ChartControlSeries_Preparesyle  
  
Private Sub series_Preparesyle(ByVal sender As Object, ByVal args As  
ChartPrepareStyleInfoEventArgs)  
Dim series As ChartSeries = TryCast(sender, ChartSeries)  
If series IsNot Nothing Then
```

```
'Condition to select members (data points) who made 100 % quota in sales
If ((series.Points(args.Index).YValues(0) / 150) * 100) >= 100
Then
    args.Style.Interior = New Syncfusion.Drawing.BrushInfo(Syncfusion.Drawing.GradientStyle.Horizontal,
1, System.Drawing.Color.DarkGreen, System.Drawing.Color.LightYellow)
End If
End If
End Sub
```



Figure 369: Green columns indicate employees who met 100 percent of their Quota

#### 4.19.5 SeriesInCompatible Event

When the Chart has completed updating the series and finds out that series are incompatible, this event will be raised.

[C#]

```
private static void chartControl1_SeriesInCompatible(object sender,
EventArgs e)
{
    Console.WriteLine("SeriesInCompatible event is raised");
```

```
}
```

**[VB.NET]**

```
Private Sub chartControl1_SeriesInCompatible(ByVal sender As Object,  
 ByVal e As EventArgs)  
     Console.WriteLine("SeriesInCompatible event is raised")  
 End Sub
```

## 4.19.6 LayoutCompleted Event

This event is handled every time, a resizing of chart is caused and when the chart re-renders itself. Listening to this event helps in cases where you render custom images over the chart or position custom controls over the chart.

**[C#]**

```
private static void chartControl1_LayoutCompleted(object sender,  
 EventArgs e)  
{  
     Console.WriteLine("Layout Completed event is raised");  
 }
```

**[VB.NET]**

```
Private Sub chartControl1_LayoutCompleted(ByVal sender As Object, ByVal  
 e As EventArgs)  
     Console.WriteLine("Layout Completed event is raised")  
 End Sub
```

## 4.19.7 ChartAreaPaint Event

ChartAreaPaint event is discussed in [Custom Drawing](#).

### See Also

[Chart Area Bounds](#)

## 4.19.8 ChartLegendFilterItems Event

This event is discussed in detail in this topic: [Chart Legend](#).

## 4.19.9 PreChartAreaPaint Event

**PreChartAreaPaint** event is raised before the chart area is painted.

[C#]

```
this.chartControl1.PreChartAreaPaint += new  
System.Windows.Forms.PaintEventHandler(this.chartControl1_PreChartAreaP  
aint);  
  
private void chartControl1_PreChartAreaPaint(object sender,  
PaintEventArgs e)  
{  
    this.chartControl1.BackColor = Color.Yellow;  
}
```

## 4.20 Localization

Localization allows a chart to display data according to the language and culture specific to a particular country or region.

Essential Chart now supports localization; built-in resource files for specific languages can be easily added. Context menu items, exception messages, and some of the toolbar items can be localized.

### Use Case Scenario

This enables you to localize any part of the chart that has static strings in it.

### Properties

Property	Description	Type	Data Type	Reference links	Dependencies

Localize	Get or set the localization culture of Grid.	Server side	A string containing the name of the target System.Globalization.CultureInfo	NA	NA
----------	----------------------------------------------	-------------	-----------------------------------------------------------------------------	----	----

### Adding Localization to an application

1. Create your localization resource file (.resx) in the **bin > Debug** folder with the following naming convention:
  - **ChartControl.<your culture info name>.resx**

*Note: It is mandatory to follow this naming convention.*

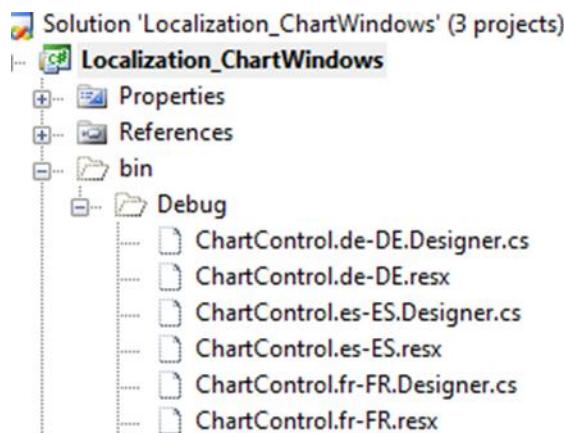
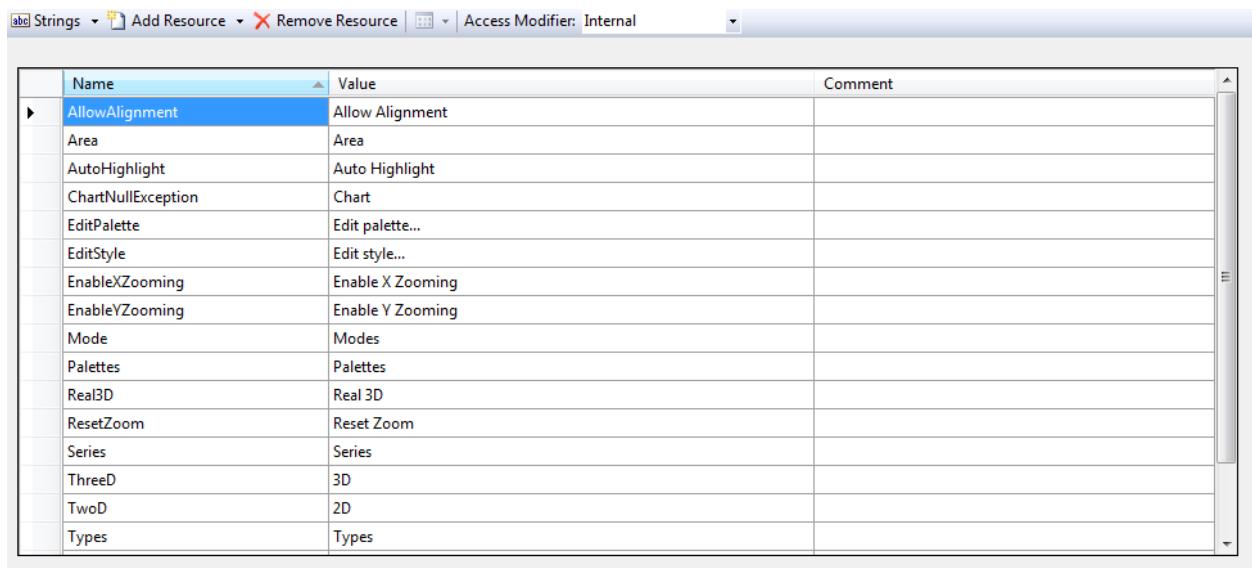


Figure 370: Resource File

2. Enter the UI name in the Name column and the equivalent term you want in the Value column of the resource file.



The screenshot shows the Windows Forms Resource Editor window. The title bar includes tabs for 'Strings', 'Add Resource', 'Remove Resource', and 'Access Modifier: Internal'. The main area is a grid table with three columns: 'Name', 'Value', and 'Comment'. The 'Name' column contains entries like 'AllowAlignment', 'Area', 'AutoHighlight', etc., and the 'Value' column contains their corresponding English terms.

Name	Value	Comment
AllowAlignment	Allow Alignment	
Area	Area	
AutoHighlight	Auto Highlight	
ChartNullException	Chart	
EditPalette	Edit palette...	
EditStyle	Edit style...	
EnableXZooming	Enable X Zooming	
EnableYZooming	Enable Y Zooming	
Mode	Modes	
Palettes	Palettes	
Real3D	Real 3D	
ResetZoom	Reset Zoom	
Series	Series	
ThreeD	3D	
TwoD	2D	
Types	Types	

Figure 371: Default English resource file

**Note:** It is mandatory to specify equivalent terms for all static element to localize the chart.

3. Specify the culture using the *Localize* property as given in the following code:

[C#]

```
this.chartControl1.Localize="de-DE";
```

[VB]

```
Me.chartControl1.Localize="de-DE"
```

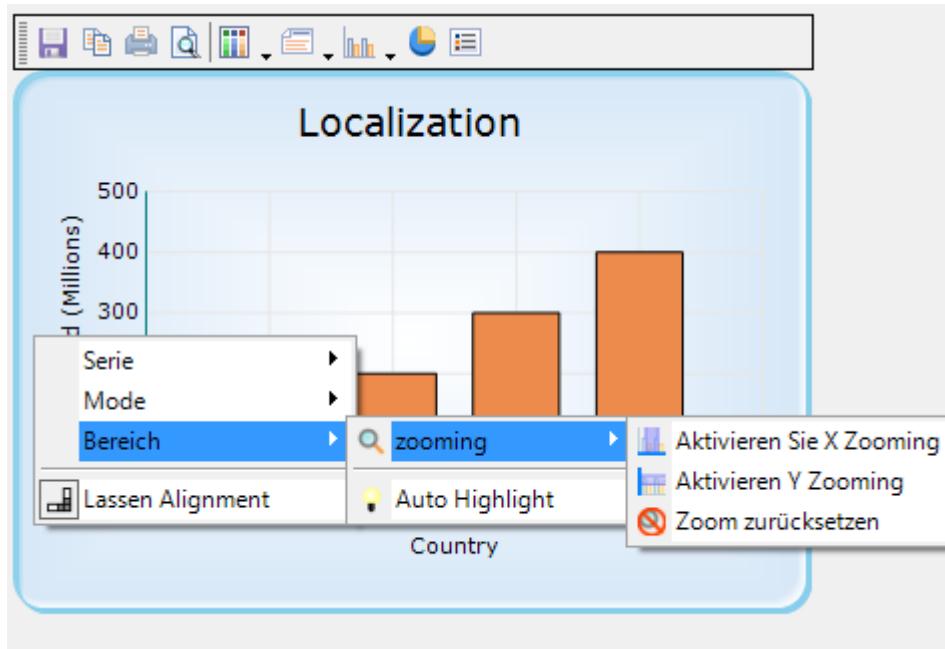


Figure 372: Localized Chart

### Sample Link

To view a sample

1. Open the Syncfusion Dashboard.
2. Select User Interface > Windows Forms.
3. Click Run Samples.
4. Navigate to **Culture Localization > Localization sample**.

You can find the resource file for the localization in English at the following location:

[ChartControl\\_Resource](#)

## 5 Frequently Asked Questions

---

This section guides you with the features of the Chart control based on specific tasks.

### 5.1 How to add custom TrendLine in Chart

TrendLines are used to draw lines in the ChartArea. They can be added to the chart using the **TrendLineAdder** class.

TrendLines can also be drawn using the Mouse Events. In this case, you will have to make use of the **Utility** class to listen to mouse events and convert them into trendlines. You can draw any number of trendlines, and can set different colors to differentiate them.

[C#]

```
// Creating Custom Points
ChartPoint ptStart = this.chart.ChartArea.GetValueByPoint(start);
ChartPoint ptEnd = this.chart.ChartArea.GetValueByPoint(end);
ChartSeries tseries = this.chart.Model.NewSeries("TrendLine",
ChartSeriesType.Line);
tseries.Points.Add(ptStart);
tseries.Points.Add(ptEnd);
this.chart.Series.Add(tseries);
tseries.LegendItem.Visible = false;

// Specify the color for the lines.
tseries.Style.Interior = new
Syncfusion.Drawing.BrushInfo(ptStart.YValues[0] < ptEnd.YValues[0] ?
Color.DarkGreen : Color.Red);
```

[VB .NET]

```
' Creating Custom Points
Dim tlineAdder As TrendLineAdder
Dim ptStart As ChartPoint = Me.chart.ChartArea.GetValueByPoint(start)
Dim ptEnd As ChartPoint =
Me.chart.ChartArea.GetValueByPoint(end_Renamed)
Dim tseries As ChartSeries = Me.chart.Model.NewSeries("TrendLine",
ChartSeriesType.Line)
tseries.Points.Add(ptStart)
```

```
tseries.Points.Add(ptEnd)
Me.chart.Series.Add(tseries)
tseries.LegendItem.Visible = False

' Specify the color for the lines.
If ptStart.YValues(0) < ptEnd.YValues(0) Then
    tseries.Style.Interior = New Syncfusion.Drawing.BrushInfo(Color.DarkGreen)
Else
    tseries.Style.Interior = New Syncfusion.Drawing.BrushInfo(Color.Red)
End If
```

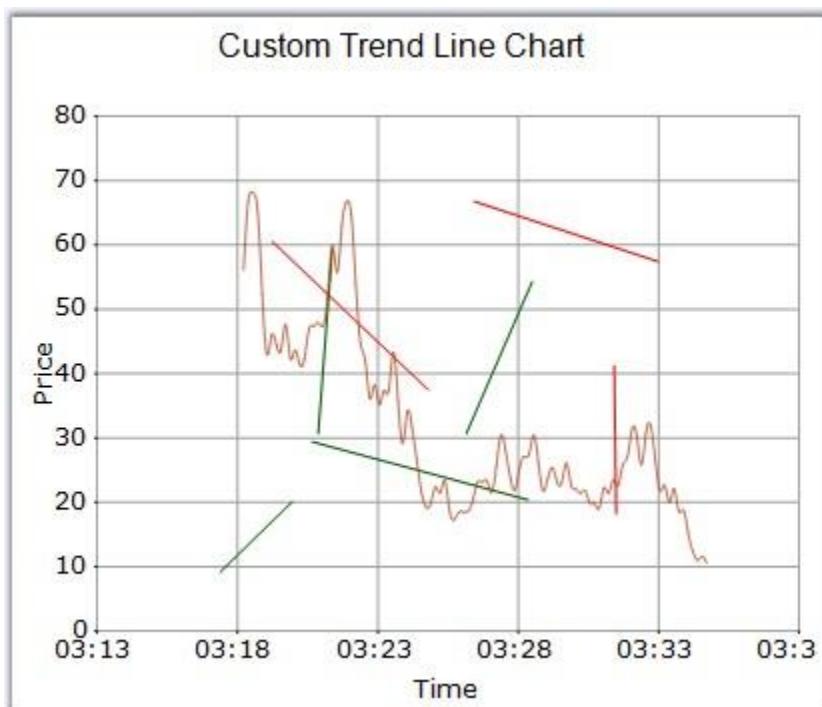


Figure 373: Custom TrendLines added to Chart

## 5.2 How to customize the data points for Chart Series

You can customize the data points by exposing the **IChartSeriesIndexedModel** interface to the series. The default Series store is an implementation of the **IChartSeriesModel**. By implementing this interface, we can set it as the underlying data.

Using the **SeriesModelImpl** property, you can set an instance of the **IChartSeriesModel**, underlying the series. Use this property to replace the instance with our own implementation.

**[C#]**

```
// Customize data points using IChartSeriesModel interface.  
series1.SeriesModelImpl = new NonIndexedModel(new double[] { 2, 5, 3,  
4, 6 }, new double[] { 10, 33, 20, 43, 12 });
```

**[VB .NET]**

```
' Customize data points using IChartSeriesModel interface.  
series1.SeriesModelImpl = New NonIndexedModel(New Double() {2, 5, 3, 4,  
6}, New Double() {10, 33, 20, 43, 12})
```

## 5.3 How to display custom tooltip over Histogram Chart columns

On Setting **ShowTooltip** property to **true**, the series name will be displayed as tooltip on the histogram chart columns by default. You can also set custom tooltip by handling **ChartRegionMouseMove** event as follows.

**[C#]**

```
private void chartControl1_ChartRegionMouseMove(object sender,  
ChartRegionEventArgs e)  
{  
    string text = null;  
    if (e.Region.IsChartPoint)  
    {  
        e.Region.ToolTip = "Tooltip " + e.Region.PointIndex.ToString();  
    }  
    else  
    {  
        text = "Not a chart Point";  
    }  
}
```

**[VB .NET]**

```
Private Sub chartControl1_ChartRegionMouseMove(ByVal sender As Object,  
ByVal e As ChartRegionEventArgs)
```

```
Dim text As String = Nothing
If e.Region.IsChartPoint Then
    e.Region.ToolTip = "Tooltip " & e.Region.PointIndex.ToString()
Else
    text = "Not a chart Point"
End If
End Sub
```

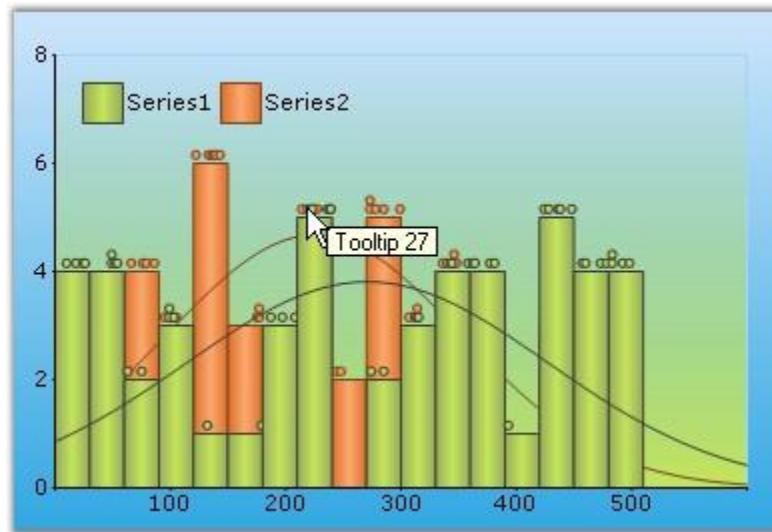


Figure 374: Custom Tooltip Displayed for Histogram Chart

#### See Also

[Chart Region Events](#), [Tooltips](#), [Histogram Chart](#)

## 5.4 How to drag the Chart Series Points at run time

You can drag the chart series points by calculating new x and y values while handling any of the **ChartRegionMouse** Events like MouseUp, MouseDown, MouseHover, MouseLeave, and so forth, on the chart. The new x and y values of the series are calculated from the mouse point, and **GetValueByPoint** which returns the x and y values of the mouse point calculated from the Chart Point.

The following code snippet must be given under the mouse event handler of the ChartRegionMouse event.

[C#]

```
private void chartControl1_ChartRegionMouseUp(object sender,
Syncfusion.Windows.Forms.Chart.ChartRegionEventArgs e)
{
    Cursor = Cursors.SizeAll;
    if (this.isDragging)
    {
        double newY =
            Math.Floor(this.chartControl1.ChartArea.GetValueByPoint(e.Point).YValues[0]);
        double newX =
            this.chartControl1.ChartArea.GetValueByPoint(e.Point).X;
        if (newY < 0 || newY >= 50 || newX < 0 || newX > 7)
            MessageBox.Show("Cannot drag outside chart bounds");
        else
        {
            this.NewYValue(newY);
            this.NewXValue(newX);
        }
        this.isDragging = false;
        this.currentRegion = null;
        this.selectedDataPoint.Y = 0;
        this.selectedDataPoint.X = 0;
        this.chartControl1.Redraw(true);
    }
    this.chartControl1.Series[0].Style.TextFormat = "{0}";
    this.chartControl1.Refresh();
}
```

[VB .NET]

```
Private Sub chartControl1_ChartRegionMouseUp(ByVal sender As Object,
ByVal e As Syncfusion.Windows.Forms.Chart.ChartRegionEventArgs)
Handles chartControl1.ChartRegionMouseUp
    Cursor = Cursors.SizeAll
    If Me.isDragging Then
        Dim newY As Double =
            Math.Floor(Me.ChartControl1.ChartArea.GetValueByPoint(e.Point).YValues(0))
        Dim newX As Double =
            Me.ChartControl1.ChartArea.GetValueByPoint(e.Point).X
        If newY < 0 OrElse newY >= 50 OrElse newX < 0 OrElse newX > 7
        Then
            MessageBox.Show("Cannot drag outside chart bounds")
        Else
            Me.NewYValue(newY)
```

```
Me.NewXValue(newX)
End If
Me.selectedDataPoint.Y = 0
Me.selectedDataPoint.X = 0
Me.isDragging = False
Me.currentRegion = Nothing
Me.ChartControl1.Redraw(True)
End If
Me.ChartControl1.Series(0).Style.TextFormat = "{0}"
Me.ChartControl1.Refresh()
End Sub
```

## 5.5 How to draw the Y-axis at the center of the X-axis

The y-axis can be drawn at any custom position using the **ChartAxisLocationType** class. This can be achieved by setting the value of the **LocationType** property of the PrimaryYAxis to **Set**.

[C#]

```
// Drawing Y-axis at the center of the X-axis.
this.chartControl1.PrimaryYAxis.LocationType =
ChartAxisLocationType.Set;
this.chartControl1.PrimaryYAxis.Location = new PointF(300, 352);
```

[VB.NET]

```
// Drawing Y-axis at the center of the X-axis.
Me.chartControl1.PrimaryYAxis.LocationType=ChartAxisLocationType.Set
Me.chartControl1.PrimaryYAxis.Location = New PointF(300, 352)
```

## 5.6 How to export the Chart Points into a CSV file

Creating a **CSV** (comma separated values) of the chart points simply involves parsing through the chart series and writing out the chart points into the FileStream. The code for this is provided below for your convenience.

[C#]

```
foreach(ChartSeries series in this.chartControl1.Series)
{
    string seriesName = series.Name;
```

```
int pointCount = series.Points.Count;
string seriesType = series.Type.ToString();

for(int p = 0; p < pointCount; p++)
{
    ChartPoint point = series.Points[p];

    string yvaluescsv = String.Empty;
    int count = point.YValues.Length;
    for(int i = 0; i < count ; i++)
    {
        yvaluescsv += point.YValues[i];
        if(i != count-1)
            yvaluescsv += comma;
    }
    csvLine = seriesName + "-" + seriesType + comma + point.X +
comma + yvaluescsv;
    csvContent += csvLine + "\r\n";
}
}

// Using stream writer class the chart points are exported. Create an
// instance of the stream writer class.
System.IO.StreamWriter file = new System.IO.StreamWriter(fileName);

// Write the datapoints into the file.
file.WriteLine(csvContent);
file.Close();
```

**[VB.NET]**

```
For Each series In Me.chartControl1.Series
    Dim seriesName As String = series.Name
    Dim pointCount As Integer = series.Points.Count
    Dim seriesType As String = series.Type.ToString()
    Dim p As Integer
    For p = 0 To pointCount - 1
        Dim point As ChartPoint = series.Points(p)
        Dim yvaluescsv As String = String.Empty
        Dim count As Integer = point.YValues.Length
        Dim i As Integer
        For i = 0 To count - 1
            yvaluescsv += point.YValues(i).ToString()
            If i <> count - 1 Then
                yvaluescsv += comma
            End If
```

```

    Next i
        csvLine = seriesName + "-" + seriesType + comma +
point.X.ToString() + comma + yvaluescsv
        csvContent += csvLine + vbCr + vbLf
    Next p
Next series

' Using stream write class the chart points are exported. Create an
instance of the stream writer class.
Dim file As New System.IO.StreamWriter(fileName)

' Write the datapoints into the file.
file.WriteLine(csvContent)
file.Close()

```

## 5.7 How to filter particular set of data points in the Chart Series

Data is filtered on a series-by-series basis. When series data points are filtered, they can be either removed from the Series Points collection or marked as **Empty**.

Also, you can reflect changes in data by filtering a particular set of data points using the **Grouping Engine**. While using the grouping engine, the main data source for the whole engine can be set. The TableDescriptor will pick up the ItemProperties from the SourceList, and table will be initialized at run time with records from the list. Using the **RecordFilterDescriptor** class, you can filter the chart point values by comparing it against a given constant value. The filtered points will be added to the series.

[C#]

```

// Generating Series
ChartSeries series =this.chartControll .Model.NewSeries ("Filter
Series",ChartSeriesType.Column );
series.Text=series.Name;
list.Clear();
for(int i=0;i<10;i++)
{
a[i]=r.Next(300,500);
series.Points.Add(i,a[i]);
list.Add(new Data(i, a[i]));
}
this.chartControll.Series.Add(series);

```

```
// Bind it to the model
Engine group=new Engine();
group.SetSourceList (list);
ExpressionFieldDescriptor exp = new ExpressionFieldDescriptor();
exp.Expression = "[Y] > "+this.textBox1.Text.ToString();
RecordFilterDescriptor rfd = new
RecordFilterDescriptor(exp.Expression);
group.TableDescriptor.RecordFilters.Add (rfd);
System.Diagnostics.Trace.WriteLine("Filtered Record Count:" +
group.Table.FilteredRecords.Count);
System.Diagnostics.Trace.WriteLine("Values greater than 30:");

// Filtering Data
this.chartControl1.Series[0].Points.Clear();
int j = 0;
foreach(Record rec in group.Table.FilteredRecords)
{
    string b=rec.GetData().ToString();
    System.Diagnostics.Trace.WriteLine(b);
    this.chartControl1.Series[0].Points.Add(j,Convert.ToDouble(b));
    j++;
}
this.label2.Text = "Number of Filtered points:
"+group.Table.FilteredRecords.Count.ToString();
```

**[VB.NET]**

```
' Generating Series
Dim series As ChartSeries = Me.chartControl1.Model.NewSeries("Filter
Series", ChartSeriesType.Column)
series.Text=series.Name
list.Clear()
For i As Integer = 0 To 9
a(i)=r.Next(300,500)
series.Points.Add(i,a(i))
list.Add(New Data(i, a(i)))
Next i
Me.chartControl1.Series.Add(series)

' Bind it to the model
Dim group As Engine = New Engine()
group.SetSourceList (list)
Dim exp As ExpressionFieldDescriptor = New ExpressionFieldDescriptor()
exp.Expression = "[Y] > " & Me.textBox1.Text.ToString()
Dim rfd As RecordFilterDescriptor = New
RecordFilterDescriptor(exp.Expression)
```

```
group.TableDescriptor.RecordFilters.Add (rfd)
System.Diagnostics.Trace.WriteLine("Filtered Record Count:" &
group.Table.FilteredRecords.Count)
System.Diagnostics.Trace.WriteLine("Values greater than 30:")

' Filtering Data
Me.chartControl1.Series(0).Points.Clear()
Dim j As Integer = 0
For Each rec As Record In group.Table.FilteredRecords
Dim b As String = rec.GetData().ToString()
System.Diagnostics.Trace.WriteLine(b)
Me.chartControl1.Series(0).Points.Add(j,Convert.ToDouble(b))
j += 1
Next rec
Me.label2.Text = "Number of Filtered points: " &
group.Table.FilteredRecords.Count.ToString()
```

## 5.8 How to hide the Chart ZoomButton

Syncfusion Chart provides a way to access the ZoomOutButton through the ScrollBar instance. Inorder to hide this Zoom button, if **Visible** property is set to **false**, ZoomButton will be disabled, but there will be an empty space. So instead of setting **Visible** property, we can set the ZoomButton size to be **0**.

### [C#]

```
this.chartControl1.GetVScrollBar(this.chartControl1.PrimaryYAxis).ZoomButton.Size = new Size(0,0);
this.chartControl1.GetHScrollBar(this.chartControl1.PrimaryXAxis).ZoomButton.Size = new Size(0, 0);
```

### [VB .NET]

```
Me.chartControl1.GetVScrollBar(Me.chartControl1.PrimaryYAxis).ZoomButton.Size = New Size(0,0)
Me.chartControl1.GetHScrollBar(Me.chartControl1.PrimaryXAxis).ZoomButton.Size = New Size(0, 0)
```

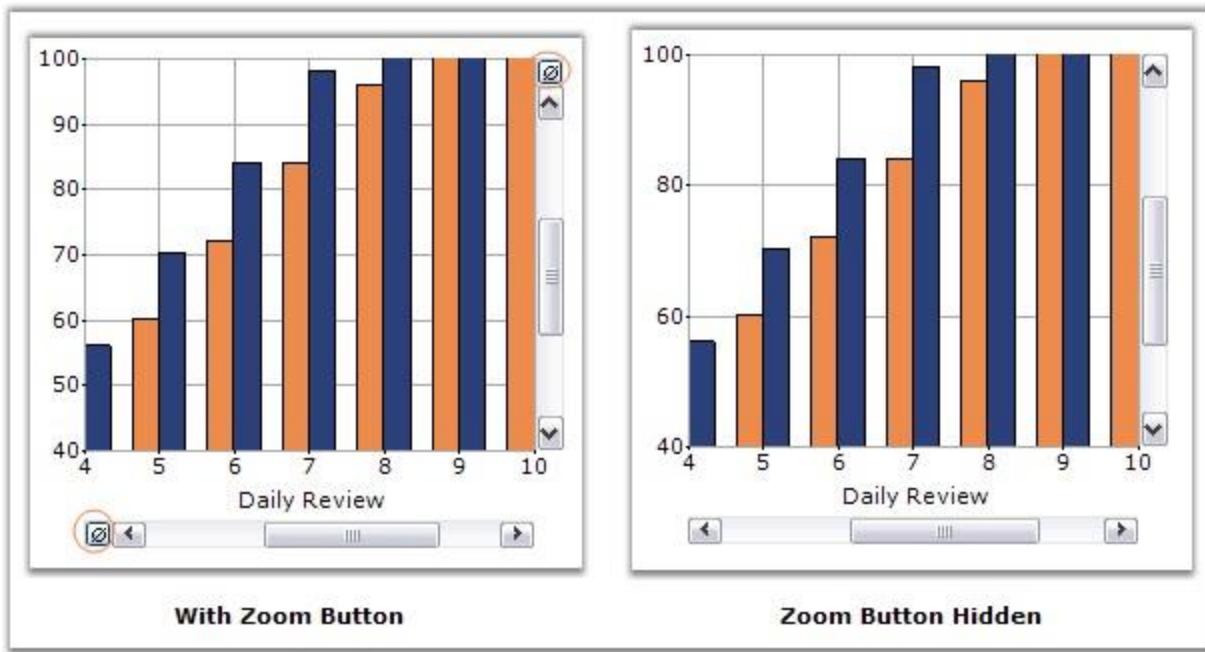


Figure 375: Hiding the Chart Zoom Button

This setting will be useful, if you need to display the scrollbar, without ZoomingCancel operation, or if you need to change the backcolor and other properties, as ZoomButton is derived from the Button control.

#### See Also

[Zooming and Scrolling](#)

## 5.9 How to show or hide the tabs in the series style dialog in a Chart

The tabs in the series style dialog can be shown or hidden using the below properties settings.

[C#]

```
this.chartControl1.StyleDialogOptions.ShowBorderTab = true;
this.chartControl1.StyleDialogOptions.ShowTextTab = false;
this.chartControl1.StyleDialogOptions.ShowFancyToolTipsTab = false;
this.chartControl1.StyleDialogOptions.ShowInteriorTab = false;
this.chartControl1.StyleDialogOptions.ShowShadowTab = false;
this.chartControl1.StyleDialogOptions.ShowSymbolTab = false;
```

[VB.NET]

```
Me.chartControl1.StyleDialogOptions.ShowBorderTab = True  
Me.chartControl1.StyleDialogOptions.ShowTextTab = False  
Me.chartControl1.StyleDialogOptions.ShowFancyToolTipsTab = False  
Me.chartControl1.StyleDialogOptions.ShowInteriorTab = False  
Me.chartControl1.StyleDialogOptions.ShowShadowTab = False  
Me.chartControl1.StyleDialogOptions.ShowSymbolTab = False
```

## 5.10 How to display the Chart Area alone

This can be achieved by setting the **Legend.Visible** property of ChartControl to **False**, **ElementsSpacing** property of ChartControl to **Zero**, and **Text** property of ChartControl to an Empty String.

[C#]

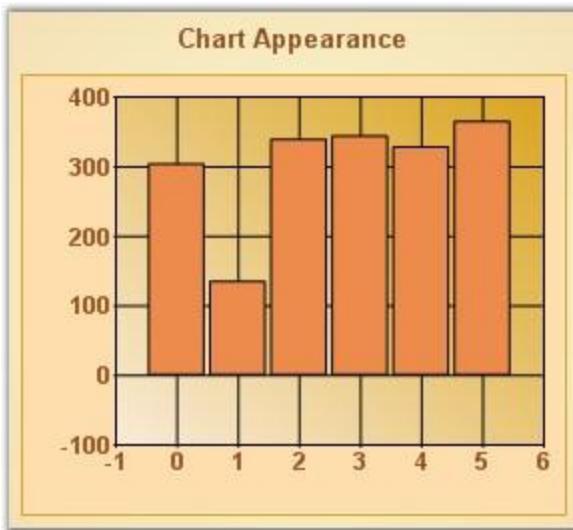
```
this.chartControl1.Text = "";  
this.chartControl1.Legend.Visible = false;  
this.chartControl1.ElementsSpacing = 0;
```

[VB.NET]

```
Me.chartControl1.Text = ""  
Me.chartControl1.Legend.Visible = False  
Me.chartControl1.ElementsSpacing = 0
```

## 5.11 How to get back to the gradient appearance of the Chart Series

The default appearance of the chart series is as shown in the image below.



*Figure 376: Chart with Flat Look and Feel*

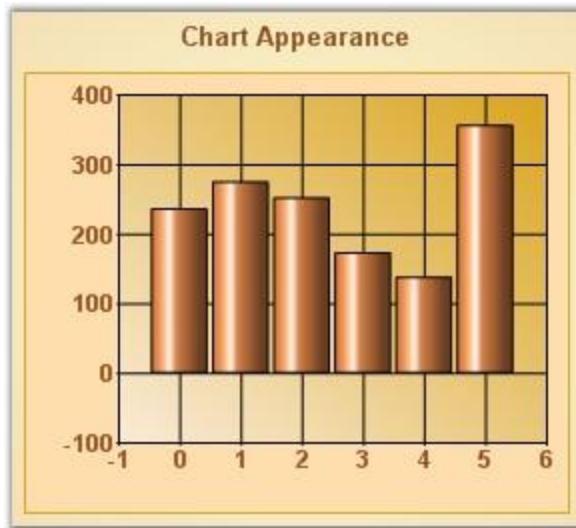
To get the gradient appearance, we need to set the **ChartControl.Model.ColorModel.AllowGradient** to **true**. By default this is set to **false**.

**[C#]**

```
//Sets the Gradient look and feel.  
this.chartControll.Model.ColorModel.AllowGradient = true;
```

**[VB.NET]**

```
'Sets the Gradient look and feel.  
Me.chartControll.Model.ColorModel.AllowGradient = True
```



*Figure 377: Chart with Gradient Look and Feel*



**Note:** We can also use `ChartControl.AllowGradientPalette` property to enable or disable gradient effect for chart series. By default it set to false.

## 5.12 How to implement DrillDown charts

DrillDown charts can essentially be implemented by listening to the click events in the chart and either replacing the current visible chart with another chart that has drill-down information or reinitializing the chart with new drill-down information.

The **ChartRegionClick** event will let you listen to the user clicking on the data points in a chart.

The sample at "[My Documents\Syncfusion\EssentialStudio\Version Number\Windows\Chart.Windows\Samples\2.0\User Interaction\Chart Drill Down](#)" illustrates the second approach.

## 5.13 How to provide input data of DateTime type

The Start Date and Time can be expressed using an instance of the **DateTime** class. If you want to add days, the **AddDays()** method can be used along with that instance. **AddHours()** and **AddMinutes()** can be used for adding any number of hours and minutes.

[C#]

```
DateTime start = new DateTime(2006, 11, 1);
ChartSeries series = this.chartControll1.Model.NewSeries("");
series.Points.Add(start.AddDays(7), 363);
series.Points.Add(start.AddDays(14), 417);
```

**[VB.NET]**

```
Dim start As DateTime = New DateTime(2006, 11, 1)
ChartSeries series = Me.chartControll1.Model.NewSeries("")
series.Points.Add(start.AddDays(7), 363)
series.Points.Add(start.AddDays(14), 417)
```

## 5.14 How to set the Color Palette for a Chart

**ChartColorPalette.Color** property can be used to specify the color palettes for the **Chart.ColorPalette** class. Apart from specifying predefined palettes, you can specify your own palette colors using the *Custom* style in the **ChartColorPalette**.

**[C#]**

```
// Specify a custom color.
this.chartControll1.CustomPalette = new System.Drawing.Color[]
{System.Drawing.Color.FromArgb(((int)((byte)(255))),
((int)((byte)(203))), ((int)((byte)(216))), System.Drawing.Color.FromArgb(((int)((byte)(222))),
((int)((byte)(209))), ((int)((byte)(248))), System.Drawing.Color.FromArgb(((int)((byte)(250))),
((int)((byte)(155))), ((int)((byte)(155))))};
this.chartControll1.Palette = ChartColorPalette.Custom;
```

**[VB.NET]**

```
' Specify a custom color.
Me.chartControll1.CustomPalette = New System.Drawing.Color()
{System.Drawing.Color.FromArgb(CType(Fix(CType(255))), CType(Fix(CType(203))), CType(Fix(CType(216))), System.Drawing.Color.FromArgb(CType(Fix(CType(222))), CType(Fix(CType(209))), CType(Fix(CType(248)))), System.Drawing.Color.FromArgb(CType(Fix(CType(250))), CType(Fix(CType(155))), CType(Fix(CType(155)))))}
Me.chartControll1.Palette = ChartColorPalette.Custom
```

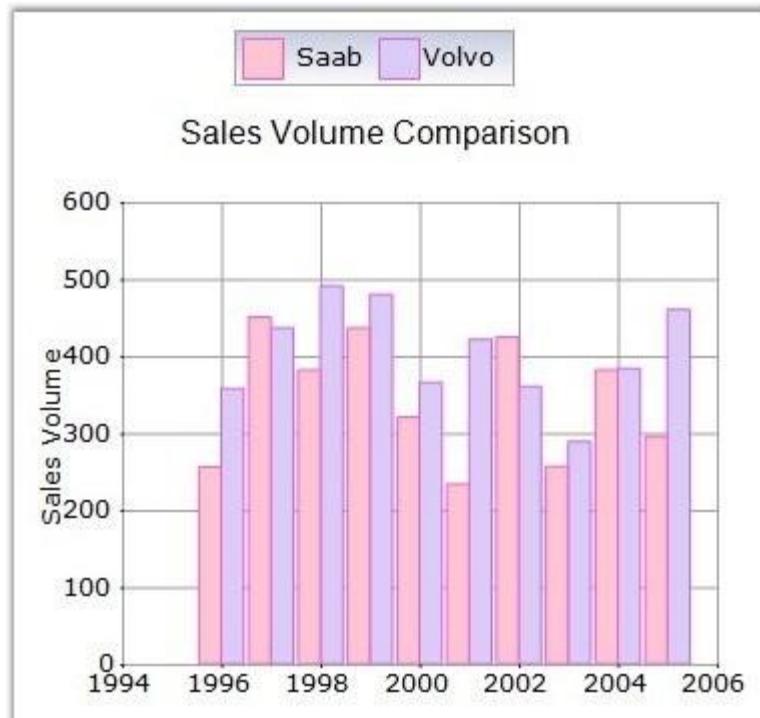


Figure 378: Chart with customized Color Palette

## 5.15 How to specify the position for a Floating Legend

When the **LegendPosition** property of the **ChartControl** is set to **ChartDock.Floating**, the position of the legend defaults to the top-right corner of the **ChartArea**. Once this is done, you can specify the coordinates via the **Legend.Location** property of the **ChartLegend**.

### [C#]

```
this.chartControl1.LegendPosition = ChartDock.Floating;
this.chartControl1.Legend.Location = new Point(20,20);
```

### [VB .NET]

```
Me.ChartControl1.LegendPosition = ChartDock.Floating
Me.ChartControl1.Legend.Location = New Point(20,20)
```

### See Also

[Chart Legend](#)

## 5.16 How to find value, maximum value and minimum value of the data points

Essential Chart has **FindValue**, **FindMaximumValue** and **FindMinimumValue** methods that can return the corresponding chart data point values depending upon the parameter(s) passed to these methods.

All these methods are overloaded.

### Find Value

Method	Description
FindValue(Double)	It should return the first chart point in the collection that has a specified first Y-value. The search always should start at the beginning of the collection.
FindValue(Double, String)	It should return the first chart point in the collection with the specified X or Y-value.
FindValue(Double, String, index )	It should return the first chart point with the specified X or Y-value, and should start the search at the specified index.
FindValue(Double, String, Index , Index )	It should return the first chart point with the specified X or Y-value, and should start and end the search at the specified index.

#### [C#]

```
dbl = Int64.Parse(txBxValue.Text);
ChartPoint dp1 = this.chartControl1.Series[0].Summary.FindValue(dbl);
```

#### [VB .NET]

```
dbl = Int64.Parse(txBxValue.Text)
Dim dp1 As ChartPoint =
Me.chartControl1.Series(0).Summary.FindValue(dbl)
```

### FindMaximumValue

Method	Description
--------	-------------

FindMaxValue()	It should return the first chart point in the collection with a maximum first Y-value. The search always should start at the beginning of the collection.
FindMaxValue(String)	It should return the first chart point in the collection with the maximum specified value. The search always should start at the beginning of the collection.
FindMaxValue(String, index)	It should return the first chart point with a maximum value. The search should start at the specified index.
FindMaxValue(String, Index , Index)	It should return the first chart point with a maximum value. The search should start and end at the specified index.

**[C#]**

```
String str = txBxString.Text;
staIndx = Int32.Parse(txBxIndex.Text);
endIndx = Int32.Parse(textBox1.Text);
ChartPoint dp4 = this.chartControl1.Series[0].Summary.FindMinValue(str,
ref staIndx, endIndx);
```

**[VB .NET]**

```
Dim str As String = txBxString.Text
staIndx = Int32.Parse(txBxIndex.Text)
endIndx = Int32.Parse(textBox1.Text)
Dim dp4 As ChartPoint =
Me.chartControl1.Series(0).Summary.FindMinValue(str, ref staIndx,
endIndx)
```

**FindMinimumvalue**

Method	Description
FindMinValue()	It should return the first chart point in the collection that has a first Y-value that is equal to the series' minimum Y1 value. The search always should start at the beginning of the collection.
FindMinValue(String)	It should return the first chart point in the collection that has an X or Y-value that is equal to the series' minimum value. The search always should start at the beginning of the collection.
FindMinValue(String, index)	It should return the first Chart point with a minimum value. The

	search should start at the specified index.
FindMinValue(String, Index , Index)	It should return the first Chart point with a minimum value. The search should start and end at the specified index.

**[C#]**

```
String str = txBxString.Text;
staIndx = Int32.Parse(txBxIndex.Text);
endIndx = Int32.Parse(textBox1.Text);
ChartPoint dp4 = this.chartControl1.Series[0].Summary.FindMaxValue(str,
ref staIndx, endIndx);
```

**[VB .NET]**

```
Dim str As String = txBxString.Text
staIndx = Int32.Parse(txBxIndex.Text)
endIndx = Int32.Parse(textBox1.Text)
Dim dp4 As ChartPoint =
Me.chartControl1.Series(0).Summary.FindMaxValue(str, staIndx, endIndx)
```

## 5.17 How to Print a Chart in Multiple Pages

To print a Chart in multiple page, use the *PrintPage* event to specify the range of X and Y axis. Specify the minium and maximum value in the *Range* property of the axis you want to divide. Set the *HasMorePages* property to true in the events before specifying the range values and set this to false after chart default maximum value. The following code illustrates this:

**[C#]**

```
this.chartControl1.PrintDocument.PrintPage += new
System.Drawing.Printing.PrintPageEventHandler(PrintDocument_PrintPage);

//PrintPage Event

void PrintDocument_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
{
if (textBox1.Text == "")
```

```
textBox1.Text = "20";

//Set the HasMorePages property to true for dividing the chart into Multiple
//pages

e.HasMorePages = true;

this.chartControl1.PrimaryXAxis.LabelIntersectAction =
ChartLabelIntersectAction.Wrap;

if (mx == 0.0 && mi == 0.0)
{
    mx = Convert.ToDouble(textBox1.Text); //Initializing max and min range
    mi = 0;
}

//Get the Color mode

bool grayScale = this.chartControl1.PrintDocument.ColorMode ==
ChartPrintColorMode.GrayScale;
bool toolBarVisibility = this.chartControl1.ShowToolbar;
if (!this.chartControl1.PrintDocument.PrintToolBar)
{
    this.chartControl1.ShowToolbar = false;
}

if (m_currentAction.Value == PrintAction.PrintToPrinter
    && this.chartControl1.PrintDocument.ColorMode ==
ChartPrintColorMode.CheckPrinter)
{
    grayScale = this.chartControl1.PrintDocument.PrinterSettings.SupportsColor;
}

//Check the color mode of print

if (!grayScale)
{
    //Assigning the initial values of max and min to chartcontrol's maximum and
    //minimum values

    this.chartControl1.ChartArea.PrimaryXAxis.Range.Min = mi;
    this.chartControl1.ChartArea.PrimaryXAxis.Range.Max = mx;
    this.chartControl1.ChartArea.PrimaryXAxis.Range.Interval =
(this.chartControl1.ChartArea.PrimaryXAxis.Range.Max -
this.chartControl1.ChartArea.PrimaryXAxis.Range.Min) /
this.chartControl1.ChartArea.PrimaryXAxis.Range.NumberOfIntervals;
```

```
//Modifying the maximum and minimum values
mi = mx;
mx = mx + Convert.ToDouble(textBox1.Text);
GraphicsContainer container = BeginTransform(e.Graphics);
e.Graphics.ResetTransform();
//Call the Draw method to draw the chart
this.chartControl1.Draw(e.Graphics, e.MarginBounds);
EndTransform(e.Graphics, container);
}

//For Grayscale mode of print
else if (grayScale)
{
ChartStyleInfo[] tempStyles = new
ChartStyleInfo[this.chartControl1.Series.Count];
Array ps = System.Enum.GetValues(typeof(PatternStyle));
Array ds = System.Enum.GetValues(typeof(DashStyle));
for (int i = 0; i < this.chartControl1.Series.Count; i++)
{
tempStyles[i] = new ChartStyleInfo();
tempStyles[i].CopyFrom(this.chartControl1.Series[i].StylesImpl.Style);
this.chartControl1.Series[i].Style.Interior = new
BrushInfo((PatternStyle)ps.GetValue(i % ps.Length), Color.Black,
Color.White);
this.chartControl1.Series[i].Style.Border.MakeCopy(tempStyles[i],
this.chartControl1.Series[i].Style.Border.Sip);
this.chartControl1.Series[i].Style.Border.Color = Color.Black;
this.chartControl1.Series[i].Style.Border.DashStyle =
(DashStyle)ds.GetValue(i % ds.Length);
//Checking the charttype
if (this.chartControl1.Series[i].Type == ChartSeriesType.Line ||
this.chartControl1.Series[i].Type == ChartSeriesType.Spline ||
this.chartControl1.Series[i].Type == ChartSeriesType.StepLine ||
this.chartControl1.Series[i].Type == ChartSeriesType.RotatedSpline)
{
this.chartControl1.Series[i].Style.Interior = new
BrushInfo((PatternStyle)ps.GetValue(i % ps.Length), Color.White,
Color.Black);
if (this.chartControl1.Series3D || this.chartControl1.ChartInterior.BackColor
```

```
== Color.Black)
{
    this.chartControl1.Series[i].Style.Interior = new
    BrushInfo((PatternStyle)ps.GetValue(i % ps.Length), Color.Black,
    Color.White);
    this.chartControl1.Series[i].Style.Border.Color = Color.Black;
}
}
}

GraphicsContainer container = BeginTransform(e.Graphics);
e.Graphics.ResetTransform();
using (Image img = new Bitmap(e.MarginBounds.Width, e.MarginBounds.Height))
{
    using (Graphics g = Graphics.FromImage(img))
    {
        //Assigning the initial values of max and min to chartcontrol maximum and
        minimum values

        this.chartControl1.ChartArea.PrimaryXAxis.Range.Min = mi;
        this.chartControl1.ChartArea.PrimaryXAxis.Range.Max = mx;
        this.chartControl1.ChartArea.PrimaryXAxis.Range.Interval =
        (this.chartControl1.ChartArea.PrimaryXAxis.Range.Max -
        this.chartControl1.ChartArea.PrimaryXAxis.Range.Min) /
        this.chartControl1.ChartArea.PrimaryXAxis.Range.NumberOfIntervals;
        //Modifying the maximum and minimum values
        mi = mx;
        mx = mx + Convert.ToDouble(textBox1.Text);
        IntPtr hdc = g.GetHdc();
        Stream stream = new MemoryStream();
        Metafile mf = new Metafile(stream, hdc);
        //Call the Draw method to draw the chart
        this.chartControl1.Draw(mf, img.Size);
        DrawingUtils.DrawGrayedImage(e.Graphics, mf, e.MarginBounds, new
        Rectangle(Point.Empty, img.Size));
        g.ReleaseHdc(hdc);
        g.Dispose();
    }
}
```

```
mf.Dispose();
}
}

EndTransform(e.Graphics, container);
for (int i = 0; i < this.chartControl1.Series.Count; i++)
{
this.chartControl1.Series[i].StylesImpl.Style.ResetInterior();
this.chartControl1.Series[i].StylesImpl.Style.ResetBorder();
this.chartControl1.Series[i].StylesImpl.Style.CopyFrom(tempStyles[i]);
}
}

//Checking Toolbar functionality of print
if (!this.chartControl1.PrintDocument.PrintToolBar)
{
this.chartControl1.ShowToolbar = toolBarVisibility;
}

// Redraws the chart
this.chartControl1.Redraw(true);

//Maximum exceeds the default range means reset the HasMorePages property.
if (mx > end)
e.hasMorePages = false;

//Finally resets the maximum and minimum value for default chartcontrol
this.chartControl1.ChartArea.PrimaryXAxis.Range.Min = start;
this.chartControl1.ChartArea.PrimaryXAxis.Range.Max = end;
this.chartControl1.ChartArea.PrimaryXAxis.Range.Interval = Intervel;
}
```

[VB]

```
Me.chartControl1.PrintDocument.PrintPage += New
System.Drawing.Printing.PrintPageEventHandler(PrintDocument_PrintPage)

''' / PrintPage Event
```

```
Private Sub PrintDocument PrintPage(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs)
If textBox1.Text = "" Then
    textBox1.Text = "20"
End If
'/' Set the HasMorePages property to true for dividing the chart into Multiple pages

e.HasMorePages = True
Me.chartControl1.PrimaryXAxis.LabelIntersectAction =
    ChartLabelIntersectAction.Wrap
If mx = 0.0 AndAlso mi = 0.0 Then
    '"/Initializing max and min range values

    mx = Convert.ToDouble(textBox1.Text)
    mi = 0
End If

'"/Get the Color mode
Dim grayScale As Boolean = Me.chartControl1.PrintDocument.ColorMode =
    ChartPrintColorMode.GrayScale
Dim toolBarVisibility As Boolean = Me.chartControl1.ShowToolbar
If Not Me.chartControl1.PrintDocument.PrintToolBar Then
    Me.chartControl1.ShowToolbar = False
End If
If m_currectAction.Value = PrintAction.PrintToPrinter AndAlso
    Me.chartControl1.PrintDocument.ColorMode =
        ChartPrintColorMode.CheckPrinter Then
    grayScale =
        Me.chartControl1.PrintDocument.PrinterSettings.SupportsColor
End If

'"/Check the color mode of print
If Not grayScale Then
    '"/Assigning the initial values of max and min to chartcontrol maximum and '"/minimum values
```

```
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Min = mi
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Max = mx
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Interval =
(Me.chartControl1.ChartArea.PrimaryXAxis.Range.Max -
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Min) /
Me.chartControl1.ChartArea.PrimaryXAxis.Range.NumberOfIntervals

'''/Modifying the maximum and minimum values
mi = mx
mx = mx + Convert.ToDouble(textBox1.Text)
Dim container As GraphicsContainer = BeginTransform(e.Graphics)
e.Graphics.ResetTransform()
'''/Call the Draw method to draw the chart

Me.chartControl1.Draw(e.Graphics, e.MarginBounds)
EndTransform(e.Graphics, container)
'''/For Grayscale mode of print
ElseIf grayScale Then
Dim tempStyles As ChartStyleInfo() = New
ChartStyleInfo(Me.chartControl1.Series.Count - 1) {}
Dim ps As Array = System.[Enum].GetValues(GetType(PatternStyle))
Dim ds As Array = System.[Enum].GetValues(GetType(DashStyle))
For i As Integer = 0 To Me.chartControl1.Series.Count - 1
    tempStyles(i) = New ChartStyleInfo()
tempStyles(i).CopyFrom(Me.chartControl1.Series(i).StylesImpl.Style)
Me.chartControl1.Series(i).Style.Interior = New
BrushInfo(DirectCast(ps.GetValue(i Mod ps.Length), PatternStyle),
Color.Black, Color.White)
Me.chartControl1.Series(i).Style.Border.MakeCopy(tempStyles(i),
Me.chartControl1.Series(i).Style.Border.Sip)
Me.chartControl1.Series(i).Style.Border.Color = Color.Black
Me.chartControl1.Series(i).Style.Border.DashStyle =
DirectCast(ds.GetValue(i Mod ds.Length), DashStyle)
'''/Checking the charttype
If Me.chartControl1.Series(i).Type = ChartSeriesType.Line OrElse
Me.chartControl1.Series(i).Type = ChartSeriesType.Spline OrElse
Me.chartControl1.Series(i).Type = ChartSeriesType.StepLine OrElse
Me.chartControl1.Series(i).Type = ChartSeriesType.RotatedSpline Then
```

```
Me.chartControl1.Series(i).Style.Interior = New
BrushInfo(DirectCast(ps.GetValue(i Mod ps.Length), PatternStyle),
Color.White, Color.Black)

If Me.chartControl1.Series3D OrElse
Me.chartControl1.ChartInterior.BackColor = Color.Black Then

Me.chartControl1.Series(i).Style.Interior = New
BrushInfo(DirectCast(ps.GetValue(i Mod ps.Length), PatternStyle),
Color.Black, Color.White)

Me.chartControl1.Series(i).Style.Border.Color = Color.Black

End If

End If

Next

Dim container As GraphicsContainer = BeginTransform(e.Graphics)
e.Graphics.ResetTransform()

Using img As Image = New Bitmap(e.MarginBounds.Width,
e.MarginBounds.Height)

Using g As Graphics = Graphics.FromImage(img)

'''Assigning the initial values of max and min to chartcontrol maximum
and '''minimum values

Me.chartControl1.ChartArea.PrimaryXAxis.Range.Min = mi
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Max = mx
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Interval =
(Me.chartControl1.ChartArea.PrimaryXAxis.Range.Max -
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Min) /
Me.chartControl1.ChartArea.PrimaryXAxis.Range.NumberOfIntervals

'''Modifying the maximum and minimum values

mi = mx
mx = mx + Convert.ToDouble(textBox1.Text)
Dim hdc As IntPtr = g.GetHdc()
Dim stream As Stream = New MemoryStream()
Dim mf As New Metafile(stream, hdc)

'''Call the Draw method to draw the chart
Me.chartControl1.Draw(mf, img.Size)
DrawingUtils.DrawGrayedImage(e.Graphics, mf, e.MarginBounds, New
```

```
Rectangle(Point.Empty, img.Size))
g.ReleaseHdc(hdc)
g.Dispose()
mf.Dispose()

End Using
End Using

EndTransform(e.Graphics, container)

For i As Integer = 0 To Me.chartControl1.Series.Count - 1

Me.chartControl1.Series(i).StylesImpl.Style.ResetInterior()

Me.chartControl1.Series(i).StylesImpl.Style.ResetBorder()

Me.chartControl1.Series(i).StylesImpl.Style.CopyFrom(tempStyles(i))

Next

End If

'''/Checking Toolbar functionality of print

If Not Me.chartControl1.PrintDocument.PrintToolBar Then
Me.chartControl1.ShowToolbar = toolBarVisibility
End If

'''/Redraws the chart
Me.chartControl1.Redraw(True)
/Reset the HasMorePages property to false when it exceeds the charts
default maximum.

If mx > [end] Then
e.HasMorePages = False
End If

//Finally resets the maximum and minimum value for default chartcontrol

Me.chartControl1.ChartArea.PrimaryXAxis.Range.Min = start
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Max = [end]
Me.chartControl1.ChartArea.PrimaryXAxis.Range.Interval = Interval
End Sub
```



**Notes:** The initial page will be printed based on the specified minimum and maximum value. In further pages minimum value will be the maximum value of previous page and maximum will be the sum of current page minimum value and the specified maximum value in the Range.

## Events

*Table 5: Event Table*

Event	Description	Arguments	Type	Reference links
PrintPage	This event will be triggered when the chart is printed.	PrintPageEventArgs	Server side	NA

## Sample Link

To view a sample:

4. Open the **Syncfusion Dashboard**.
5. Click the **Windows Forms** drop-down list and select **Run Locally Installed Samples**.
6. Navigate to **Chart Samples -->Print -->Multiple Page Printing**.

## **Index**

[

[C#] 608

**1**

1 Overview 11

1.1 Introduction to Essential Chart 11

1.2 Prerequisites and Compatibility 14

1.3 Documentation 14

**2**

2 Installation and Deployment 16

2.1 Installation 16

2.2 Sample and Location 16

2.3 Deployment Requirements 19

**3**

3 Getting Started 20

3.1 Control Structure 20

3.2 Creating a simple Chart 22

**4**

4 Concepts and Features 26

4.1 Chart Wizard 26

4.1.1 Chart Type 28

4.1.2 Series 29

4.1.3 Appearance 33

4.1.4 Axes 35

4.1.5 Points 37

4.1.6 Toolbar 38

4.1.7 Legend 41

4.10 Chart Appearance 491

4.10.1 Background Colors 491

4.10.10 Chart Skins 516

4.10.2 Background Image 494

4.10.3 Border and Margins 497

4.10.4 Foreground Settings 502

4.10.5 Multiple Chart Titles 504

4.10.6 Custom Drawing 509

4.10.7 Watermark Support 511

4.10.8 Interlaced Grid Background 513

4.10.9 Minor Grid Lines 514

4.11 Realtime 525

4.12 Statistical Formulas 526

4.12.1 Basic Statistical Formulas 526

4.12.1.1 Anova Test 527

4.12.1.10 Z-Test 547

4.12.1.2 Correlation 531

4.12.1.3 Covariance 533

4.12.1.4 F-Test 535

4.12.1.5 Mean 536

4.12.1.6 Median 537

4.12.1.7 Standard Deviation 538

4.12.1.8 T-Tests 539

4.12.1.8.1 TTest with Equal Variances 540

4.12.1.8.2 T-Test with UnEqual Variance 542

4.12.1.8.3 TTest Paired 544

4.12.1.9 Variance 546

4.12.2 Utility Functions 549

4.12.2.1 Beta Function 550

4.12.2.10 Inverse Error Function 561

4.12.2.11 Inverse F Cumulative Distribution 561

4.12.2.12 Inverse Normal Distribution 562

4.12.2.13 Normal Distribution 563

4.12.2.14 Normal Distribution Density 565

4.12.2.15 Inverse T Cumulative Distribution 567

4.12.2.16 TCumulative Distribution 567

4.12.2.2 Beta Cumulative Distribution 551

4.12.2.3 Binomial Coefficient 553

4.12.2.4 Inverse Beta Cumulative Distribution 554

4.12.2.5 Error Function 555

- 4.12.2.6 Factorial 556
- 4.12.2.7 F Cumulative Distribution 557
- 4.12.2.8 Gamma Function 558
- 4.12.2.9 Gamma Cumulative Distribution 560
- 4.13 Importing 569
  - 4.13.1 Importing a CSV file 569
  - 4.13.2 Import Data from XML to a Chart 570
  - 4.13.3 Import Data from Excel to Chart 570
- 4.14 Exporting 571
  - 4.14.1 Exporting as an Image 571
  - 4.14.2 Exporting to Word Doc 574
  - 4.14.3 Exporting to Grid 577
  - 4.14.4 Exporting to Excel 581
  - 4.14.5 Exporting to PDF 584
- 4.15 Design time Features 586
  - 4.15.1 Chart Templates 586
  - 4.15.2 Tasks Window 590
- 4.16 Printing and Print Preview 594
- 4.17 Data Manipulation 597
- 4.18 Hit Testing 600
  - 4.18.1 Chart Coordinates by Point 600
  - 4.18.2 LegendItem By Point 601
  - 4.18.3 Chart Area Bounds 603
- 4.19 Chart control events 605
  - 4.19.1 Chart Region Events 606
  - 4.19.2 VisibleRangeChanged Event 609
  - 4.19.3 ChartFormatAxisLabel Event 610
  - 4.19.4 PrepareStyle Event 610
  - 4.19.5 SeriesInCompatible Event 611
  - 4.19.6 LayoutCompleted Event 612
  - 4.19.7 ChartAreaPaint Event 612
  - 4.19.8 ChartLegendFilterItems Event 613
  - 4.19.9 PreChartAreaPaint Event 613
- 4.2 Chart Data 43
  - 4.2.1 Binding a DataSet to the Chart 44
  - 4.2.2 Implementing Custom Data Binding Interfaces 47
  - 4.2.3 Chart Data Binding with IEnumerables 50
  - 4.2.4 Data Binding in Chart Through Chart Wizard 54
- 4.20 Localization 613
- 4.3 Improving Performance 67
- 4.4 Chart Types 70
  - 4.4.1 Line Charts 72
    - 4.4.1.1 Line Chart 72
    - 4.4.1.2 Spline Chart 74
    - 4.4.1.3 Rotated Spline Chart 76
    - 4.4.1.4 Step Line Chart 78
  - 4.4.10 Combination Chart 153
  - 4.4.11 Heat Map Charts 155
  - 4.4.12 Stacking Charts 158
  - 4.4.13 Step Charts 158
  - 4.4.14 Sparkline 159
- 4.4.2 Bar Charts 80
  - 4.4.2.1 Bar Chart 80
  - 4.4.2.2 Stacking Bar Chart 83
  - 4.4.2.3 StackedBar100 Chart 85
  - 4.4.2.4 Gantt Chart 87
- 4.4.2.5 Histogram Chart 88
- 4.4.2.6 Tornado Chart 91
- 4.4.3 Column Charts 93
  - 4.4.3.1 Column Chart 93
  - 4.4.3.2 Column Range Chart 95
  - 4.4.3.3 Stacking Column Chart 97
  - 4.4.3.4 Stacked Column100 Chart 99
- 4.4.4 Area Charts 101
  - 4.4.4.1 Area Chart 102
  - 4.4.4.2 Spline Area Chart 103
  - 4.4.4.3 Stacking Area Chart 106
  - 4.4.4.4 StackedArea100 Chart 108

4.4.4.5 Step Area Chart 110	4.5.1.2 Border 179
4.4.4.6 Range Area Chart 112	4.5.1.20 EnableAreaToolTip 217
4.4.5 Accumulation Charts 114	4.5.1.21 ErrorBarsSymbolShape 218
4.4.5.1 Funnel Chart 114	4.5.1.22 ExplodedAll 220
4.4.5.2 Pyramid Chart 118	4.5.1.23 ExplodedIndex 221
4.4.6 XY Charts (Bubble and Scatter) 121	4.5.1.24 ExplosionOffset 222
4.4.6.1 Scatter Chart 122	4.5.1.25 FancyToolTip 224
4.4.6.2 Bubble Chart 124	4.5.1.26 FigureBase 225
4.4.7 Financial Charts 126	4.5.1.27 FillMode 230
4.4.7.1 Candle Chart 127	4.5.1.28 FunnelMode 232
4.4.7.2 Hi Lo Chart 129	4.5.1.29 Font 235
4.4.7.3 Hi Lo Open Close Chart 130	4.5.1.3 BubbleType 182
4.4.7.4 Kagi Chart 132	4.5.1.30 GanttDrawMode 237
4.4.7.5 Point and Figure Chart 135	4.5.1.31 GapRatio 239
4.4.7.6 Renko Chart 137	4.5.1.32 Gradient 241
4.4.7.7 Three Line Break Chart 140	4.5.1.33 HeightBox 243
4.4.7.8 Box And Whisker Chart 142	4.5.1.34 HeightByAreaDepth 245
4.4.8 Pie Chart 145	4.5.1.35 HeightCoefficient 247
4.4.8.1 Doughnut Chart 147	4.5.1.36 HighlightInterior 248
4.4.9 Polar And Radar Chart 148	4.5.1.37 HitTestRadius 251
4.4.9.1 Polar Chart 149	4.5.1.38 ImageIndex 253
4.4.9.2 Radar Chart 151	4.5.1.39 Images 255
4.5 Chart Series 168	4.5.1.4 ColumnDrawMode 184
4.5.1 Series Customization 171	4.5.1.40 InnerRadius 257
4.5.1.1 AngleOffset 177	4.5.1.41 Interior 259
4.5.1.10 DisplayShadow 198	4.5.1.42 LabelPlacement 262
4.5.1.11 DisplayText 200	4.5.1.43 LabelStyle 265
4.5.1.12 DoughnutCoefficient 203	4.5.1.44 LegendItem 267
4.5.1.13 DrawColumnSeparatingLines 204	4.5.1.45 LightAngle 268
4.5.1.14 DrawErrorBars 206	4.5.1.46 LightColor 270
4.5.1.15 DrawHistogramNormalDistribution 210	4.5.1.47 Name 271
4.5.1.16 DrawSeriesNameInDepth 211	4.5.1.48 NumberOfHistogramIntervals 273
4.5.1.17 DropSeriesPoints 213	4.5.1.49 OpenCloseDrawMode 274
4.5.1.18 ElementBorders 214	4.5.1.5 ColumnWidthMode 187
4.5.1.19 EnablePhongStyle 216	4.5.1.50 OptimizePiePointPositions 278

4.5.1.51 PhongAlpha 281	4.5.1.83 TextColor 344
4.5.1.52 PieType 283	4.5.1.84 TextFormat 346
4.5.1.53 PieWithSameRadius 285	4.5.1.85 TextOffset 348
4.5.1.54 PointsToolTipFormat 286	4.5.1.86 TextOrientation 349
4.5.1.55 PointWidth 286	4.5.1.87 ToolTip 352
4.5.1.56 PriceDownColor 288	4.5.1.88 ToolTipFormat 355
4.5.1.57 PriceUpColor 290	4.5.1.89 Visible 358
4.5.1.58 PyramidMode 292	4.5.1.9 DarkLightPower 196
4.5.1.59 Radar Type 293	4.5.1.90 VisibleAllPies 360
4.5.1.6 ColumnFixedWidth 191	4.5.1.91 XType 362
4.5.1.60 RadarStyle 295	4.5.1.92 YType 363
4.5.1.61 RelatedPoints 297	4.5.1.93 ZOrder 365
4.5.1.62 ReversalAmount 300	4.5.2 Data Point Labels, Tooltips and Symbols 368
4.5.1.63 Rotate 302	4.5.3 Custom Points 369
4.5.1.64 ScatterConnectType 304	4.5.3.1 Custom Point in Multiple Axes 374
4.5.1.65 ScatterSplineTension 306	4.5.4 Empty Points 375
4.5.1.66 SeriesToolTipFormat 308	4.6 Chart Axes 379
4.5.1.67 ShadingMode 310	4.6.1 Indexed X Values 380
4.5.1.68 ShadowInterior 311	4.6.10 Axis Ticks 415
4.5.1.69 ShadowOffset 314	4.6.11 3-D Related 418
4.5.1.7 ColumnType 193	4.6.12 Chart Grid Lines 422
4.5.1.70 ShowDataBindLabels 316	4.6.13 Chart StripLines 424
4.5.1.71 ShowHistogramDataPoints 319	4.6.14 Chart Breaks 428
4.5.1.72 ShowTicks 321	4.6.15 Axis Crossing Support 431
4.5.1.73 SmartLabels 323	4.6.16 Axis Label Placement 432
4.5.1.74 Spacing 325	4.6.2 Inverted Axis 382
4.5.1.75 SpacingBetweenSeries 327	4.6.3 Opposed Axis 383
4.5.1.76 SpacingBetweenPoints 329	4.6.4 Multiple Axes 384
4.5.1.77 Stacking Group 330	4.6.5 Axis Value Type 389
4.5.1.78 StepItem.Inverted 332	4.6.6 Axis Range and Intervals 390
4.5.1.79 Summary 334	4.6.7 Axis Dimensions 394
4.5.1.8 ColorsMode 195	4.6.8 Axis Labels 397
4.5.1.80 Symbol 336	4.6.8.1 Axis Label Text Formatting, Appearance and Positioning 397
4.5.1.81 Text (Series) 340	4.6.8.2 Customizing Label Text 399
4.5.1.82 Text (Style) 342	

4.6.8.3 Intersecting Labels 405	5.2 How to customize the data points for Chart Series 618
4.6.8.4 Grouping Labels 406	5.3 How to display custom tooltip over Histogram Chart columns 619
4.6.8.5 Tooltip Support for ChartAxisLabels 408	5.4 How to drag the Chart Series Points at run time 620
4.6.9 Axis Title 412	5.5 How to draw the Y-axis at the center of the X-axis 622
4.7 Chart Area 437	5.6 How to export the Chart Points into a CSV file 622
4.8 Chart Legend and Legend Items 438	5.7 How to filter particular set of data points in the Chart Series 624
4.8.1 ChartLegend 439	5.8 How to hide the Chart ZoomButton 626
4.8.2 ChartLegendItem 443	5.9 How to show or hide the tabs in the series style dialog in a Chart 627
4.8.3 Customizing LegendItem Image 450	
4.9 Runtime Features 458	
4.9.1 Zooming and Scrolling 458	
4.9.2 Toolbars 464	
4.9.2.1 Toolbar Properties 467	
4.9.2.2 Appearance 469	
4.9.3 Context Menu 472	
4.9.4 Interactive Features 474	
4.9.4.1 Drawing Interactive Cursor Separately – Either Horizontally or Vertically or Both 478	
4.9.4.2 ChartInteractiveCursor Support for Chart Area 484	
4.9.5 ToolTips 486	
<b>5</b>	
5 Frequently Asked Questions 617	
5.1 How to add custom TrendLine in Chart 617	
5.10 How to display the Chart Area alone 628	
5.11 How to get back to the gradient appearance of the Chart Series 628	
5.12 How to implement DrillDown charts 630	
5.13 How to provide input data of DateTime type 630	
5.14 How to set the Color Palette for a Chart 631	
5.15 How to specify the position for a Floating Legend 632	
5.16 How to find value, maximum value and minimum value of the data points 633	
5.17 How to Print a Chart in Multiple Pages 635	