

LegendBox Tutorials

Table of Contents

1.	Box Tutorial	1-5
2.	LegendBox Tutorial Basic	6-9
3.	LegendBox Advanced	10-15

1 Box Tutorial

Default Box

A **Box** object is the foundation (base class) of box like objects such as **annotations**, **legendBoxes**, and title boxes. The chart control has a **DefaultBox** property that is a container for box properties that will be propagated to all other objects on the chart that derive from Box including annotations, legends, and title boxes. The default box corner related settings will also propagate to the chart areas.

Example: This code will make all box shadows red.

```
[C#]
Chart.DefaultBox.Shadow.Color = Red;

[Visual Basic]
Chart.DefaultBox.Shadow.Color = Red
```

Styling

Box styling consists of a background, the outline (Line), corner types, headers, and shadows.

Background

A box's background is accessed through its **Background** property, which contains properties that effect the background's visual appearance:

Box.Background.___ = ___

Example: The following code sample changes the box's background color to red.

```
[C#]
Box.Background.Color = Color.Red;

[Visual Basic]
Box.Background.Color = Color.Red
```

See the **Background** class for more information.



See sample: [BoxStyling.aspx](#)

Shadow

The shadow a box casts can also be modified. Options include color, depth, and whether the shadow is solid or soft, meaning, the edges are blurred for added realism.

```
[C#]
Box.Shadow.Color = Color.Orange;

[Visual Basic]
Box.Shadow.Color = Color.Orange
```

Outline

A Line object is used to draw the outline of a box. This enables you to specify many visual properties such as line width, color, and dash style.

Example: Changes the outline dash style.

```
[C#]
Box.Outline.DashStyle = DashStyle.Dash;

[Visual Basic]
Box.Outline.DashStyle = DashStyle.Dash
```

See also: **Line Class, Chart Lines ('Lines' in the on-line documentation)**

Corners

Each of the four corners of a box can be individually styled. This code demonstrates how to specify a

rounded top left corner.

```
[C#]
Box.CornerTopLeft = BoxCorner.Round;

[Visual Basic]
Box.CornerTopLeft = BoxCorner.Round
```

All four corners can be automatically set using the **DefaultCorner** property. This will also change the property value of each individual corner.

```
[C#]
Box.DefaultCorner = BoxCorner.Cut;

[Visual Basic]
Box.DefaultCorner = BoxCorner.Cut
```

Corner Size

The corner size can also be specified through the `Box.CornerRadius` property. It is given in pixels.

⚠️ A legendbox corner size is limited to 0-8 unless the legendbox padding is set to a higher value that will prevent text clipping. Other boxes that use a setting higher than 8 will automatically add padding to fit text inside it.

💡 Tip: Today's styling trends dictate that using a unique corner on opposite ends of a box while keeping the adjacent corners intact is visually pleasing. For Example `TopLeft:Cut` and `BottomRight: Cut` while `TopRight` and `BottomLeft` setting is: `Square`.

Padding

The padding property specifies the distance from the box edges to its content similar to HTML cell padding. The `LegendBox`, which inherits from `Box` also uses the padding property to determine spacing between legend entries.

```
[C#]
Box.Padding = 5;

[Visual Basic]
Box.Padding = 5
```

📌 Note: Custom corners do not effect a box's padding, hence, additional padding may be required when content is too close to a styled corner like `BoxCorner.Cut`.

Box Headers

Box headers is a new feature in version 4.4. It allows a header label and background style on top of any box such as `TileBox`, `LegendBox`, or `Annotation`. The header is activated simply by specifying a `Box.HeaderLabel.Text` string. If there is no text available, the header is not visible. The following example adds a header to the legend box with a different background.



```
[C#]
Chart.LegendBox.HeaderLabel.Text = "Legend Box";
Chart.LegendBox.HeaderBackground.Color = Color.Green;

[Visual Basic]
Chart.LegendBox.HeaderLabel.Text = "Legend Box"
Chart.LegendBox.HeaderBackground.Color = Color.Green
```

💡 See Sample: Features: `BoxHeaders.aspx`

Box Header Styling

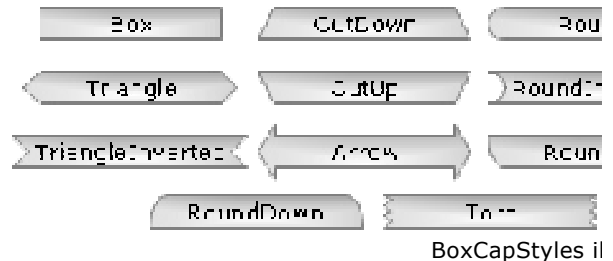
Version 6.1 introduces a much more advanced header feature set. It provides an API that enables styling headers to emulate many different visualizations. **Box.Header** is now its own object containing all the header related properties.

Header Cap Styles

The header left and right sides can be styled to change their shape. The header object contains two properties; **StartCap**, and **EndCap** which refers to the left and right sides, respectively. Both properties can use any style specified and will render the header accordingly. There are currently 13 box cap styles available allowing 169 different combinations. The styles are described by the **BoxCapStyle** enumeration, and the code below shows how they can be set.

```
[C#]
Box.Header.StartCap = BoxCapStyle.Triangle;
Box.Header.EndCap = BoxCapStyle.Triangle;

[Visual Basic]
Box.Header.StartCap = BoxCapStyle.Triangle
Box.Header.EndCap = BoxCapStyle.Triangle
```



Ribbon Caps

Ribbon caps can be set with **BoxCapStyle.RibbonUp/RibbonDown** options.

Ribbon caps behavior is slightly more complex as the ends must extend to the back of the parent box. Ribbons may alter their angles in order to reach the back of the parent box.

The ribbons can render differently depending on the header positions. Figure 1b shows headers with the same RibbonDown settings. The banner extends outside the left and right sides while the tab does not and the resulting rendering is different.



RibbonDown

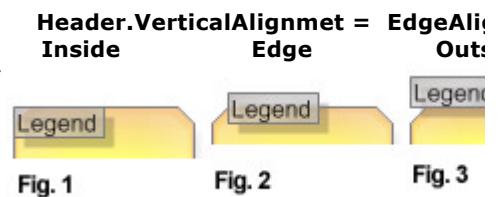
If the ribbon setting is not applicable, it will be omitted. For example when the header is above the box and RibbonUp is set, the ribbon cannot reach the parent box moving up so no cap will be rendered.

Vertical Alignment

Header boxes can be positioned vertically in relation to the parent box edge. The **EdgeAlignment** enumeration specifies this positioning with Inside/Outside/Edge. Figure 1-3 demonstrates the positions when used with the **Box.Header.VerticalAlignment** property.

```
[C#]
Box1.Header.VerticalAlignment = EdgeAlignment.Inside;
Box2.Header.VerticalAlignment = EdgeAlignment.Edge;
Box3.Header.VerticalAlignment = EdgeAlignment.Outside;

[Visual Basic]
Box1.Header.VerticalAlignment = EdgeAlignment.Inside
Box2.Header.VerticalAlignment = EdgeAlignment.Edge
Box3.Header.VerticalAlignment = EdgeAlignment.Outside
```



Horizontal Alignment

The header object has two properties; **StartAlignment**, and **EndAlignment**. They dictate how the left and right sides of the header are aligned to the parent box and header label, but they do not affect the label alignment to the parent box.

Box.Header.Label.Alignment

The label alignment will dictate the label's position across the entire parent box regardless of any header start and end alignments settings. The label positions are shown below in Figure 4.

```
[C#]
Box.Header.Label.Alignment = StringAlignment.Near;

[Visual Basic]
Box.Header.Label.Alignment = StringAlignment.Near
```



Fig. 4
Header label StringAlignment

options illustrated.

Box.Header.StartAlignment/EndAlignment

The Header.**StartAlignment** and Header.**EndAlignment** properties use the same **EdgeAlignment** enumeration as Header.[VerticalAlignment](#) property, and they behave similarly. Outside and Edge options will align based on the parent box edge. Inside alignment will align at the label alignment or box edge. The Inside setting also considers if the box sides have a cap. When a header label is aligned to the box edge with a cap, it might offset the label position so both cap and label can be within the parent box bounds. The figures below show what these settings do when used with Header.**StartAlignment**. The behavior is the same with Header.**EndAlignment**.

EdgeAlignment.Outside - The header box will extend past the parent box edge by the pixel amount specified with Box.CornerRadius.

EdgeAlignment.Edge - The header box will line up with the parent box edge. If specified, a the base of a cap is aligned with the parent box and is rendered just outside the box as shown in figure 7.

 Edge Notes

- In order to have the cap tip touch the box edge, the label must be aligned to that edge and startAlignment = Inside. (See figure 7)
- When the vertical alignment is Edge and the parent box has corners, the header will be offset by CornerSize as shown in Figure 2.

EdgeAlignment.Outside - When the label is not aligned to this edge, the header side will align with the label. When the label is aligned with this side, and the side has a cap style, the label is pushed in so the header and cap can be rendered inside the box. (See figure 7)

Header Cap Placement

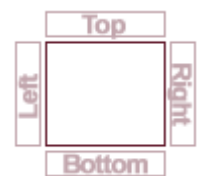
When using **BoxCapStyle.Box**, no cap is used. With other styles, a cap is drawn outside of the original box. So Box capped headers will be a bit smaller than others. The width of the caps is half the height of the header label. (See Figure 1a)

```
[C#]
Box.Header.StartAlignment = EdgeAlignment.Outside;
```

```
[Visual Basic]
Box.Header.StartAlignment = EdgeAlignment.Outside
```

Header Orientation

The property Box.Header.Orientation controls the orientation of the header in relation its parent box. Figure 9 shows how the header is oriented depending on the different orientation settings. All the other alignment settings will behave the same regardless of the orientation. The bottom orientation however orients vertically so it can be read so some settings may change to achieve the same visual. For example cap styles such as RibbonDown will have to change to RibbonUp with Orientation.Bottom. Corner orientations such as TopRight are not supported at this time.

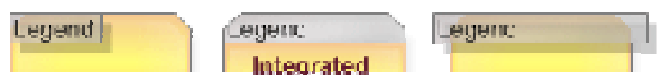


Different behavior when default settings are used.

Fig. 9

Integrated Header

When both sides of the header are aligned to Edge, and both sides use the Box cap style, the header will integrate with the parent box as the middle box in figure 8 does. These are the default settings.



Different behavior when default settings are used.

When integrated, the **header.Line** is not



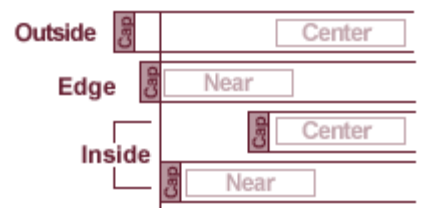
StartAlignment with a centered header label.

Fig. 5



StartAlignment with a left justified header label.

Fig. 6



StartAlignment with a start cap.

Fig. 7

used to draw an outline. Instead, the parent box outlines the header and content together. `Box.InteriorLine` is used to draw a divider line between the header and content.

Fig. 8

Other settings

Header.Line

This line object is used to draw an outline of the header box shape.

Header.Offset

This property takes a Point object and uses it to offset the header position from the original placement. It is a simple yet powerful feature that can be used to make small visual adjustments to achieve the exact look that is needed.

Header.Shadow

The header shadow defines properties of the shadow cast beneath it.

💡 To create a header visual that appears to be floating above the parent box, try using offset and shadow together.
(See floating visual below).

Visualizations

The header can emulate a number of different visuals such as banners, tabs, pointers and so on. This image shows some examples of these:



Uses

This header styling can be utilized in many scenarios to improve aesthetics of charts and to make elements more visually intuitive.

- Element Annotations (enhance callout)
- Arbitrary Annotation / Legend Header Styling

2 LegendBox Tutorial Basic

Styling

Before proceeding, see the **Box tutorial (Section 1)** for basic styling options.

Fonts:

A legend entry's font can be defined by specifying a default font for all entries.

```
[C#]
Chart.LegendBox.DefaultEntry.LabelStyle.Font = new Font("Verdana",8);

[Visual Basic]
Chart.LegendBox.DefaultEntry.LabelStyle.Font = New Font("Verdana",8)
```

See also: **Using Fonts ('Working with Fonts' in the on-line documentation)**

Colors:

Font colors can also be specified in a similar manner:

```
[C#]
Chart.LegendBox.DefaultEntry.LabelStyle.Color = Color.Black;

[Visual Basic]
Chart.LegendBox.DefaultEntry.LabelStyle.Color = Color.Black
```

See also: **Using Colors (on-line documentation)**

Template

The default columns of a legend are Name, Value, and Icon. They are represented in the **Template** property as tokens. The Template property allows reordering, exclusion of any particular column, or additional tokens that represent the same as the legend entries such as "%ElementCount".

Example: The following code reorders legend columns.

```
[C#]
Chart.LegendBox.Template = "%Icon%Value%Name";

[Visual Basic]
Chart.LegendBox.Template = "%Icon%Value%Name"
```

Example: The following code excludes the icon column.

```
[C#]
Chart.LegendBox.Template = "%Value%Name";
[Visual Basic]
Chart.LegendBox.Template = "%Value%Name"
```

Example: The following code inserts additional columns.

```
[C#]
Chart.LegendBox.Template = "%Value%Name%Icon%ElementCount%Average";
[Visual Basic]
Chart.LegendBox.Template = "%Value%Name%Icon%ElementCount%Average"
```

Text Alignment

The columns can be either icons or text. When text is used, it can be aligned either to the right or left. These alignment settings can be specified by using an array of *StringAlignment* enumerations that correspond to the above template tokens (IconValueName) respectively.

Example: The following code specifies the alignment of three columns in a legend box.

```
[C#]
Chart.LegendBox.ColumnAlignments = new StringAlignment[]{StringAlignment.Far, StringAlignm
[Visual Basic]
Chart.LegendBox.ColumnAlignments = New StringAlignment(){StringAlignment.Far, StringAlignm
```

Positioning

Orientation

To position a legend box, the `LegendBox.Position` or `LegendBox.Orientation` properties must be defined.

```
[C#]
LegendBox.Orientation = Orientation.BottomLeft;
[Visual Basic]
LegendBox.Orientation = Orientation.BottomLeft
```

The position property is an object type and accepts a legacy `LegendBoxPosition` enumeration or a `Point` object, which specifies the absolute position in pixels.

💡 Tip: Using `LegendBoxPosition` is not recommended because `Orientation` has more options.

```
[C#]
LegendBox.Position = LegendBoxPosition.Bottom;
```

```
[Visual Basic]
LegendBox.Position = LegendBoxPosition.Bottom
```

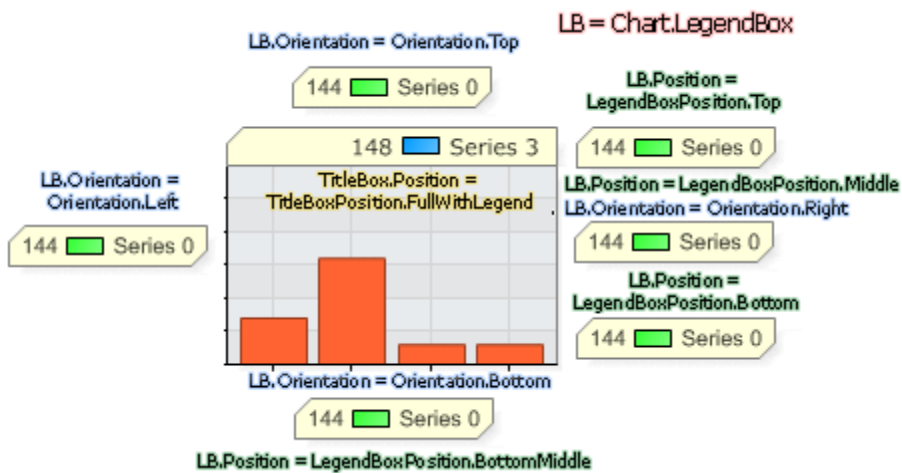


Illustration of legend box positions and corresponding properties.

Absolute Position

Using a `Point` object will specify an absolute position. Other elements on a chart will not be affected by the position and the legend will overlap objects beneath it.

```
[C#]
LegendBox.Position = new Point(200,20);
```

```
[Visual Basic]
LegendBox.Position = New Point(200,20)
```

Absolute Position with Size

The legendbox absolute position can also be specified with a size in addition to the point. This is accomplished by using a `Rectangle` object.

```
[C#]
LegendBox.Position = new Rectangle(new Point(200,20),new Size(100,100));
```

```
[Visual Basic]
LegendBox.Position = New Rectangle(New Point(200, 20), New Size(100, 100))
```

💡 Tip: When using an absolute position, it is useful to set a transparency for the box's background color so underlying objects can still be seen if necessary.

Specifying a LegendBox size

The legendbox.Position property can also take a Size object by itself. In this case, the position of the legendbox is not affected but the size specified is a static size and will not change. It will force the legend entries in the legendbox to display in the best possible way provided there is enough room for them.

```
[C#]
LegendBox.Position = new Size(100,100);
```


```
[Visual Basic]
LegendBox.Position = New Size(100, 100)
```

In Title box

Another useful option is to completely eliminate the legend box and move the entries into the title box. To accomplish this, a title box position FullWithLegend is used.

```
[C#]
Chart.TitleBox.Position = TitleBoxPosition.FullWithLegend;
```

```
[Visual Basic]
Chart.TitleBox.Position = TitleBoxPosition.FullWithLegend
```

 **Note:** The entries in a title box are positioned opposite to the alignment of the title (TitleBox.Label.Alignment). If the title is right-aligned the entries will position themselves on the left side of the title box.

Invisible Legend

To completely eliminate the legend from a chart set the Visibility of the legend to false.

```
[C#]
Chart.LegendBox.Visible = false;
```

```
[Visual Basic]
Chart.LegendBox.Visible = False
```

Legend Entries**Manipulating the content**

Entries in the legend are for most cases generated to represent each series. However, if a Series.Palette or PaletteName is specified, each element in that series will automatically insert its entry into the legend. Legend entries specific to each series and element can be accessed through their parent object. For example, an element's legend entry is accessed through:

```
myElement.LegendEntry.___ = ___
```

AxisMarker objects used with axes or elements are also automatically inserted into the legend and their entries are accessed through:

```
myAxisMarker.LegendEntry.___ = ___
```

To prevent an axis marker from entering into the legend, the entry's visibility can be turned off:

```
[C#]
myAxisMarker.LegendEntry.Visible = false;
[Visual Basic]
myAxisMarker.LegendEntry.Visible = False
```

To exclude the all series entries from a legend box the following code can be used:

```
[C#]
Chart.DefaultSeries.LegendEntry.Visible = false;
[Visual Basic]
Chart.DefaultSeries.LegendEntry.Visible = False
```

Custom Entries

Custom entries can be added to the legend by instantiating and adding them to the **ExtraEntries** property of a LegendBox.

```
[C#]
LegendEntry myEntry = new LegendEntry();
myEntry.Name = "CustomName";
myEntry.Value = "CustomValue";
Chart.LegendBox.ExtraEntries.Add(myEntry);
```

```
[Visual Basic]
Dim myEntry As New LegendEntry()
myEntry.Name = "CustomName"
myEntry.Value = "CustomValue"
Chart.LegendBox.ExtraEntries.Add(myEntry)
```

Header Entry

Each legend has a ready made header entry. These are used to describe the columns of your legend. The header entry is activated by setting the Visible property.

```
[C#]
LegendBox.HeaderEntry.Visible = true;
```

```
[Visual Basic]
LegendBox.HeaderEntry.Visible = true
```


Header text can be modified with the traditional text properties.

```
LegendBox.HeaderEntry.Name = "Series Name";
```

To change the icon text a custom attribute must be used

```
[C#]
LegendBox.HeaderEntry.CustomAttributes = "Icon=Icon Header";
```

```
[Visual Basic]
LegendBox.HeaderEntry.CustomAttributes.Add("Icon","Icon Header")
```

 Attributes will be covered further in the **advanced tutorial (Section 3)**.



More Tips:

Using your own entries.

- In some cases, it is required to use your own custom entries instead of the default ones. The best way to accomplish this is to set `Chart.DefaultSeries.LegendEntry.Visible = false;` and add your entries as shown above.
- Moving entries to the title box may save valuable space when there are few series and chart sizes are small.

The features covered in this tutorial are the most common but only scratch the capability surface of LegendBox objects.

For more advanced features go on to the **advanced tutorial (Section 3)**.

3 LegendBox Advanced

Introduction

The .netCHARTING legend box is more than just a list of icon-name pairs. It is equipped with template functionality that can detail your data in a grid style form and offers many options that will allow you to modify the box as much as you need.

Legend Positions & Layout

The legend can be positioned in three different ways. It can be placed anywhere around the chart plot areas, inside the title box along with the chart title, and it can be positioned at a specified x,y or absolute position.

Around the plot chart areas

The traditional way to position the boxes used this syntax

```
[C#]
Chart.LegendBox.Position = LegendBoxPosition.Top;

[Visual Basic]
Chart.LegendBox.Position = LegendBoxPosition.Top
```

This would specify the box is placed in the upper right corner around the plot area. A more flexible setting is also available, the equivalent of the above is:

```
[C#]
Chart.LegendBox.Orientation = Orientation.TopRight;

[Visual Basic]
Chart.LegendBox.Orientation = Orientation.TopRight
```

Orientation exposes all corners and sides available for positioning.


Absolute Positioning

The legend can be placed at any specified x,y position using the same LegendBox.Position property.

```
[C#]
Chart.LegendBox.Position = new Position(20,20);

[Visual Basic]
Chart.LegendBox.Position = New Position(20,20)
```

When this method is used, the other object on the chart will not move away to make room for the legend box, however, it may be desirable to place the legend within a ChartArea in some circumstances.

 *Sample:* LegendBoxAbsolute.aspx

In the title box

A legend can be placed in the title of a chart area by specifying to the title box that it should display the legend within it.

```
[C#]
Chart.TitleBox.Position = TitleBoxPosition.FullWithLegend;


[Visual Basic]
Chart.TitleBox.Position = TitleBoxPosition.FullWithLegend
```


When the legend entries are shown in the title box further positioning features are available. By specifying the title alignment, the legend entries will position themselves to compliment the new title's layout.

```
[C#]
Chart.TitleBox.Label.Alignment = StringAlignment.Center;

[Visual Basic]
Chart.TitleBox.Label.Alignment = StringAlignment.Center
```

The above line will center the title in response, the legend entries will center beneath it.

 NOTE: When a legend is shown in the title box, its properties are still controlled through the LegendBox object.

 Sample: LegendBoxTitle.aspx

Layout Behavior

When a legend box is located on the left or right side of a chart the default layout will yield a single column legend if it fits. With multiple legends on one side they will be stacked. This means that if a single column configuration will not fit vertically, the legends will try to increase the number of columns to fit all the entries.

Legends positioned on top and bottom will by default create multiple columns to allow the maximum vertical space for a chart.

In either arrangement, entries will list top to bottom. This behavior can be changed by setting LegendBox.ListTopToBottom to false. This setting will list the entries left to right across columns.

Multiple Chart Areas and Legends

Main Chart Area

Every extra chart area has its own legend box (ChartArea.LegendBox) , however, it is not visible by default. When this is the case the chart area's series entries are placed in the main chart legend box (Chart.LegendBox). In Figure 1, chart areas B and D send series to the main legend box. This is the default behavior.

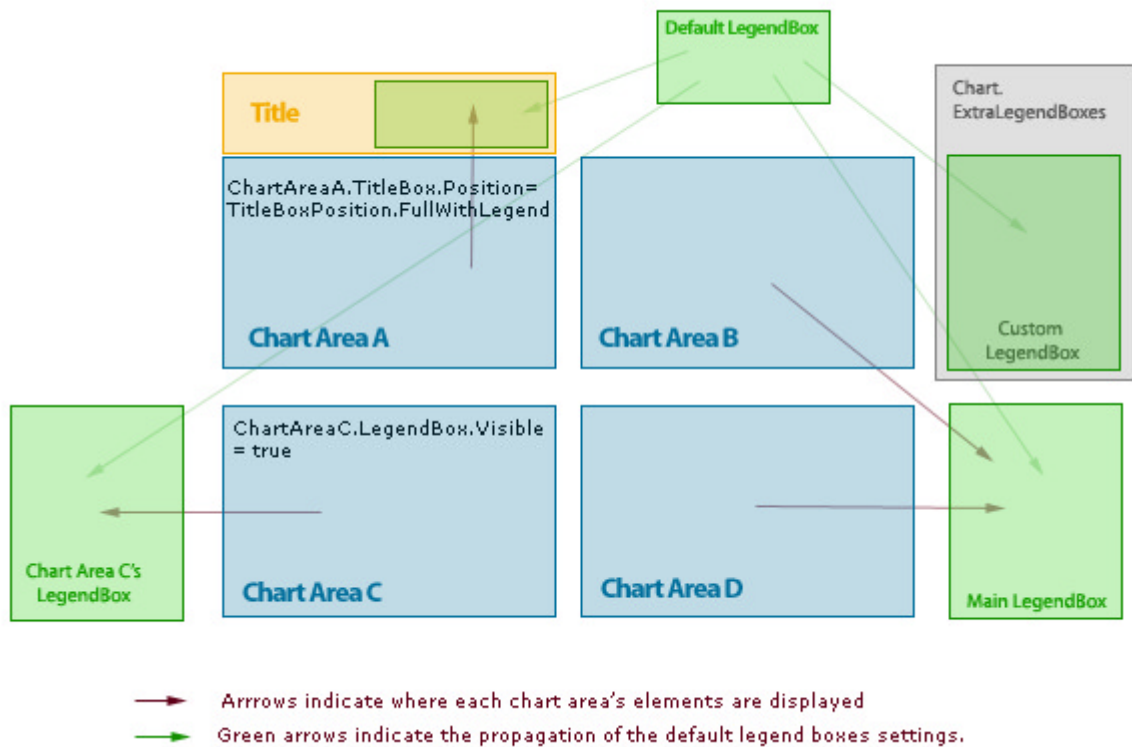



Figure 1: Illustrates how series of a particular chart area bind to legend boxes and the default legend box settings.

Personal Chart Area Legends

A chart area can place its series in a personal legend box by instantiating a legendBox object and specifying it to the chart area's LegendBox property (Figure 1: Chart Area C). A Chart area can also place its series legend entries into the title box by setting its title box's position to TitleBoxPosition.FullWithLegend (Figure 1: Chart Area A).

```
[C#]
ChartAreaA.TitleBox.Position = TitleBoxPosition.FullWithLegend;
```

```
[Visual Basic]
ChartAreaA.TitleBox.Position = TitleBoxPosition.FullWithLegend
```


 *Sample:* LegendBoxMultiple.aspx

Custom Legend Boxes

Custom legend box instances can also be added to the Chart.ExtraLegendBoxes collection. Custom entries can be added and data sources specified to populate your custom boxes.

```
[C#]
Chart.ExtraLegendBoxes.Add(new LegendBox());
```

```
[Visual Basic]
Chart.ExtraLegendBoxes.Add(New LegendBox())
```

 *Sample:* LegendBoxCustom.aspx

Default Legend Box

The default legend box (Chart.DefaultLegendBox) does not appear anywhere on the chart image but is used to quickly specify settings all boxes appearing on chart will acquire. (Figure 1: Custom LegendBox)

The Legend Box

Legend Template

The LegendBox.**Template** property takes a series of tokens which define the columns of each entry. These tokens can be tokens specific to the object the entry represents or properties of the legend entry (Name, Value, and Icon).

Consider the following code.

```
[C#]
SeriesA.DefaultElement.CustomAttributes.Add("SeriesDescription","Data Provided by Group A.
Chart.LegendBox.DefaultEntry.CustomAttributes.Add("EntryCustom","(Max: /%Maximum Min: /%Mi
Chart.LegendBox.Template = "%Name%Icon%Value%EntryCustom%SeriesDescription%YAverage";
```

```
[Visual Basic]
SeriesA.DefaultElement.CustomAttributes.Add("SeriesDescription","Data Provided by Group A.
Chart.LegendBox.DefaultEntry.CustomAttributes.Add("EntryCustom","(Max: %Maximum %Min: %Min
Chart.LegendBox.Template = "%Name%Icon%Value%EntryCustom%SeriesDescription%YAverage"
```

We add a custom attribute to a series which we want to end up in the chart’s legend box. Next we add a custom attribute named “EntryCustom” to the legend’s default entry which will propagate it to all the entries in the legend. Finally, we specify a series of tokens for the legend columns.


The following Table shows the path of each column value when the legend is rendered.


LegendBox.Template	<i>%Name</i>	<i>%Icon</i>	<i>%Value</i>	<i>%EntryCustom</i>	<i>%SeriesDescription</i>	<i>%YAverage</i>
LegendEntry source	LE.Name	LE.Marker	LE.Value	LE.CustomAttributes ("EntryCustom")	<i>Because these tokens don't apply to any legend entry source they will be passed down to the data source.</i>	
LegendEntry value/result	<i>%Name</i>	LE.Marker	<i>%YSum</i>	<i>(Max: %Maximum Min: %Minimum)</i>		
Series source	Series.Name	(Settings contribute to final legend entry marker)	Series Calculation	Series Calculation	Series.DefaultElement.CustomAttributes ("SeriesDescription")	Series Calculation
Final Result	<i>Group A</i>	LE.Marker	<i>503</i>	<i>(Max: 40 Min: 2)</i>	<i>Data Provided by Group A. 34</i>	

Table 1: Describes the path of tokens specified in LegendBox.Template

1. First the legend box column template is examined.
2. Next it looks at a particular legend entry for matches. The token values our legend entry can provide are %Name %Icon %Value and %custom where custom is the name of an entry’s custom attribute. We added this in the second line of the above sample.

- Finally the series or another object the entry represents is examined for column token matches. In this case %YAverage is a token associated with all series and %SeriesDescription a custom series attribute we added in the code above.

 Note: LegendBox.Template can only accept tokens. Other text and characters will be ignored.

 Sample: LegendBoxCustom.aspx

The Legend Entry

Default Entry


The legend box contains a default legend entry LegendBox.**DefaultEntry** property. The setting of this entry will propagate to all entries that end up in the legend box. The exception is when a setting is explicitly set for any given entry, in that case the default entry setting will be ignored.

Custom Entries (ExtraEntries)

Custom legend entries can be created and added to any legend. These may serve as headers or additional info you need to specify.

Custom Attributes

Legend entries contain a collection of custom attributes which can be used to specify further tokens to use with custom legend box column tokens. Or in a case of custom entries in a box with series entries, because the entry doesn't represent a series, custom attributes can fill in the blanks. For example ("%YSum", "n/a").

 Sample: LegendBoxCustom.aspx

Header specific features

Legend entries are also designed to serve as headers. To add a header to a legend box let's start with creating an entry and specify the header text. The legend sample in "The Legend" section will be the assumed legend we're working with.

```
[C#]
LegendEntry leHeader = new LegendEntry();
leHeader.Name = "Name";
leHeader.Value = "Sum";
```

```
[Visual Basic]
Dim leHeader As New LegendEntry()
leHeader.Name = "Name"
leHeader.Value = "Sum"
```

To specify labels for all the other column tokens they must be added to custom attributes.

```
[C#]
Le.CustomAttributes.Add("%EntryCustom", " Data Range");
Le.CustomAttributes.Add("%SeriesDescription", " Description");
Le.CustomAttributes.Add("%YAverage", "Average");
```

```
[Visual Basic]
Le.CustomAttributes.Add("%EntryCustom", " Data Range")
Le.CustomAttributes.Add("%SeriesDescription", " Description")
Le.CustomAttributes.Add("%YAverage", "Average")
```

We will also want to use text instead of the entry's marker for the icon column. This can be achieved by specifying the following custom attribute:

```
[C#]
Le.CustomAttribute("%Icon", "Icon");
```

```
[Visual Basic]
Le.CustomAttribute("%Icon", "Icon")
```

SortOrder

The header column should be at the top of the legend. This can be done by specifying a sort order for the header entry and all other entries.

```
[C#]
Le.SortOrder = 0;
LegendBox.DefaultEntry.SortOrder = 1;

[Visual Basic]
Le.SortOrder = 0
LegendBox.DefaultEntry.SortOrder = 1
```

This will ensure the header entry with the sort order of 0 will appear before all other entries which have the sort order value of 1.

Header Mode

Another way to ensure the header appears at the top is to specify the RepeatOnEachColumn header mode.

```
[C#]
Le.HeaderMode = LegendEntryHeaderMode.RepeatOnEachColumn;

[Visual Basic]
Le.HeaderMode = LegendEntryHeaderMode.RepeatOnEachColumn
```

As the name suggests, if the legend splits into multiple columns, this entry will be repeated at the top of each.

Another useful mode is StartNewColumn, when headers are used to group entries in a multi column legend. This mode will allow each group to be placed in separate columns. Such a configuration may make the legend easier to read.

The default mode of all entries is None and doesn't change the entry behavior.

Divider Line

Up to this point our header doesn't stand out in any way to indicate it is one. One way to emphasize it is by specifying a DividerLine color.

```
[C#]
Le.DividerLine.Color = Color.Black;

[Visual Basic]
Le.DividerLine.Color = Color.Black
```

This line will be drawn beneath the header entry. An extra layer of padding will be added to separate the entry and the next entry evenly from this line.

Top Padding

Single column legends that contain multiple entry groups with headers should use additional padding above each header. This will improve the visual separation of each group.

```
[C#]
Le.TopPadding = 5;

[Visual Basic]
Le.TopPadding = 5
```

The following section will demonstrate more options that may improve your headers.

 *Sample:* LegendBoxHeader.aspx

Styling

The header features section describes several styling features. Here we expand on this and tie up some loose ends.

The most fundamental legend entry styling customization is found in the LegendEntry.LabelStyle property. It allows you to set the font, its size, style and color. All entries can be affected by setting the LegendBox.DefaultEntry.LabelStyle property or each individual entry can be customized separately..

```
[C#]
Le.LabelStyle.Font = new Font("Verdana",9);
Le.LabelStyle.Color = Color.Green;
```

```
[Visual Basic]
Le.LabelStyle.Font = New Font("Verdana",9)
Le.LabelStyle.Color = Color.Green
```

Padding

The Box object which LegendBox inherits from contains a padding property. Legend boxes use this property to space entries.


Icon

A legend entry's icon can be manipulated by a number of its properties. These include

- Use3D
- SeriesType
- ShapeType
- Marker
- DashStyle
- Background

These properties specify the kind of chart element the entry icon is representing. For example, setting SeriesType.Marker gives full control of the icon to the LegendEntry.Marker property.

Now an image can be used instead of the pre defined markers.

 NOTE: When an image marker is drawn in the legend its size will be limited to twice the height of the entry's font height. If larger, it will be scaled down. Alternatively a Marker.Size value can be specified.

Populating Legends with Entries Automatically

See: **DataSource Tutorial ('Data Sources' in the on-line documentation)**

Tips & Tricks

- **Entry as a spacer**
An empty entry may be used to separate groups of header entries.
- **Divider Line Style**
Specifying a divider line with a subtle color for the default entry can create a cool effect by dividing each entry. Lowering the padding can bring the entries closer together which may look even better.
- **Control columns**
By adding entries with HeaderMode.StartNewColumn the columns can be controlled.