ComponentOne

# Gauges for WinForms

**ComponentOne, a division of GrapeCity**
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

**Website:** http://www.componentone.com
**Sales:** sales@componentone.com
**Telephone:** 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

## Trademarks



## Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

## Copying and Distribution

# Table of Contents

# Gauges for WinForms Overview

| | |
|---|---|
| **Gauges for WinForms** supports linear and radial gauges to provide an intuitive and attractive way to display information graphically. **Gauges for WinForms** provides the flexibility to create simple, practical gauges that get the job done while also supplying a multitude of advanced features to create the most eye-catching, professional-looking gauges imaginable. | **Getting Started**<br>Get started with the following topics:<br>• Key Features<br>• Quick Start<br>• Components<br>• Samples |

# Help with WinForms Edition

## Getting Started

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

## Key Features

**Gauges for WinForms** incorporates several key features, including the following:

- **Radial and Linear Gauges**

  When setting up C1Gauge, choose between radial or linear shapes. Radial gauges can be circular, helical, arc curve, clamshell, or half-circular and linear gauges can be horizontal, vertical, tilted, or free-formed.

- **Vector-based Graphics**

  All graphics are vector-based in C1Gauge unless you specify the pixel-based size. So the gauge paints itself perfectly in any size, and all inner elements resize proportionally.

- **Data-bound Gauges**

  Bind C1Gauge to a data source at design-time using standard .NET data-binding techniques. You can bind individual pointers to different data sources (up to five pointers for a single Gauge).

- **Multiple Pointers and Scales**

  There's no limit to the number of pointers and scales one gauge can hold. You can add multiple related or disjointed scales to one gauge, or overlay multiple gauges for more layout options. Scales can be non-uniform: linear or logarithmic. It's possible to create dynamic scales. You can also bind the beginning and ending of a scale to pointers.

- **Bound Ranges**

  With **Gauges for WinForms** you can create non-linear or linear shaped ranges. Customize the exact location and size of the ranges to best suit any desired look. Map colors to value thresholds to display a multi-colored range with optional gradient blending. Ranges can also be bound to the pointers for a more dramatic display. You can highlight the current value if you bind some color in the range (as well as in markers and labels) to the pointer. So you can, for example, display the progress bar using a single range object with the bound color.

- **Label Formatting**

  Apply standard or custom .NET numeric formats to all gauge labels and value indicators to display decimal places, percents, currency, and so on. The static text can appear on labels as part of the custom numeric format. You can also use the special event that gives programmatic control over the label formatting. C1Gauge can automatically rotate labels for radial gauges so that they are always most readable to users. You can specify additional rotation, change alignment, and radial/orthogonal offset to achieve interesting effects.

- **Markers**

  Markers are visual cues that can be placed at specific values on the gauge scale. These are useful for comparing the gauge value to some other predetermined value. Markers can be shapes or custom images. In the same way as labels, markers can be rotated or not rotated for radial gauges. You can apply additional rotation, change alignment and offset of gauge markers.

- **Indicators**

  You can display visual indications based upon value thresholds using bound labels and markers. Use these as state indicators in addition to or instead of ranges to visually display the value as a color or an image. For example, the background color of a marker or the marker’s image can depend on the pointer value. Or you can attach the fixed or movable label to a pointer and display the current value at the given location.

- **Custom Pointers**

Choose from the predefined pointer shapes, customize the shape, or import your own custom image to use as the pointer. Create common pointer shapes for use with multiple pointers. You can also specify the exact position of the pointer origin in radial and linear gauges. This allows you to decentralize the pointer to either side or the bottom of the gauge.

- **Off Position**

  The gauge pointer can indicate that the value is not set. If the current value is Double.NaN, then the pointer moves to the special Off **Position**. You can display a marker and a label at the off position. When using interaction features, the user can click at this position to “turn off” the gauge, or reset its value to Double.NaN.

- **Pointer Animation**

  Set the time interval so the gauge pointer will animate smoothly as the value changes. You can also decrease the frequency of redrawing for the gauge control if the source value is changed too frequently for observation.

- **Custom Gauge Appearance**

  Create any look desirable by customizing the face and cover shapes of the gauge using the rich set of style attributes available. You can even simulate a glassing effect using simple shapes. If shapes are not enough, you can add images. It's possible to apply various effects to images, such as rotation, flipping, changing the hue, saturation, lightness, and transparency.

- **Common Appearance Settings**

  The common collections of fillings, gradients, shapes, images, and so on allow the user to apply the same settings to multiple gauge items. Also, when you change these common settings, all related items will be affected at once. For example, you can create a few bound indicators that share the same mapping of images to value thresholds. This saves memory and facilitates subsequent changes.

- **UI Interaction**

  Any parts of C1Gauge can be hit-testable. It’s easy to change visual appearance of a gauge item when it becomes hot, pressed, or disabled. The visual state of gauge items can be changed smoothly using the transition effect that hides the previous state and shows the new state during the given time interval. The special events occur when the user clicks various elements or drags the gauge pointer. You may update the pointer value from these events with optional rounding to the uniform discrete set of values specified by the snap interval.

- **Composite Gauges**

  Align multiple gauges into one container using C1Gauge. Gauges can be overlapped or placed side-by-side. Since all graphics are vector-based you can resize the container control in arbitrary way. The fine-tuning settings give an ability to maintain the aspect ratio and relative position of individual gauges when resizing the container control.

- **Save and Load Layout and Appearance Settings**

  Create several views (or "skins") for a gauge or for the container control. Using views you can change the whole look of a gauge without breaking any existent scales, data, and event bindings. Views belong to a concrete gauge or **C1Gauge**. You can’t apply the same view to different gauges.

- **Save and Load XML Templates**

  Rapidly decrease development time by saving and re-using gauge templates. You can create templates for individual gauges or for the whole container control. C1Gauge also ships with several pre-designed templates to get started.

# Gauges for WinForms Quick Start

In this section you'll learn how to use the basic **Gauges for WinForms** functionality to create a simple application with a custom gauge control. This section is not intended to be a comprehensive tutorial of all features of **Gauges for WinForms**, but rather provide a quick introduction and highlight some general approaches to using the product.

In the following quick start guide, you'll create an application, add a gauge control to the application, and customize the appearance of the control.

# Step 1 of 3: Creating the Application

In this step you'll create a simple application using the C1Gauge control. You'll then customize the appearance of your application in Design view without adding any code to your project.

To begin, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select a language in the left pane, and in the templates list select **Windows Forms Application**. Enter a **Name** for your project and click **OK**. The project will be created and a form will appear.
3. Navigate to the Toolbox and double-click the **C1Gauge** item to add a **C1Gauge** control to the form. When you double-click the **C1Gauge** item, the **New Gauge Gallery** dialog box will open.
4. On the **Radial Gauges** tab, select **(empty)**.
5. Click **OK** to close the dialog box.

## What You've Accomplished

In this step you created an application and added a C1Gauge control to the form. The gauge is currently displayed with default settings; in the next step you'll further customize the appearance of the control.

# Step 2 of 3: Customizing C1Gauge

In the last step you created new application and added a C1Gauge control to the form. In this step you'll customize the gauge by using the **Gauges for WinForms** designers.

To customize the C1Gauge control, complete the following steps:

1. Double-click the **C1Gauge1** control on the form. The **Item Editor** appears.
2. Set the Maximum property to **120**. The gauge scale will now run from 0 to 120.

## Adding Tick Marks

1. Click the **ellipsis** button next to the **Decorators** property. The **Decorators Collection Editor** appears.
2. Click the drop-down list on the **Add** button and select C1GaugeMarks.
3. Expand the **Filling** node and set the Color property to **DarkGray**.
4. Set the **Interval** property to **10**. This will create tick marks at every $10^{th}$ interval. Next we will add minor tick marks.
5. Click the drop-down list on the **Add** button again and select C1GaugeMarks.
6. Set the Interval property to **2.5**.
7. Expand the **Filling** node and set the **Color** property to **DarkGray**.
8. Set the **Length** property to **5**. Next we will add tick labels.

## Adding Tick Labels

1. Click the drop-down list on the **Add** button and select C1GaugeLabels.
2. Set the Color property to **Black**.
3. Set the Interval property to **10**. This will create tick labels at every 10th interval.
4. Set the From property to **20**. This will add the labels on value 20 and higher.
5. Click **OK** to close the **Decorators Collection Editor**.

## Customize the Pointer, Cap, and Caption

1. In the **Item Editor**, expand the **Filling** node within the **Pointer** node, and set the **Color** property to **Black**.
2. Expand the **Filling** node within the **Cap** node, and set the Color property to **DarkGray**.
3. Click the **ellipsis** button next to **FaceShapes**. The **FaceShapes Collection Editor** opens.
4. Click the **Add** drop-down arrow and select **C1GaugeCaption**.
5. Enter **C1Gauge** next to the Text property.
6. Set the Color property to **Red**.
7. Enter **.9** next to the CenterPointY property. This will move the text down the gauge.
8. Click **OK** to close the **FaceShapes Collection Editor** and click **OK** again to close the **Item Editor**.

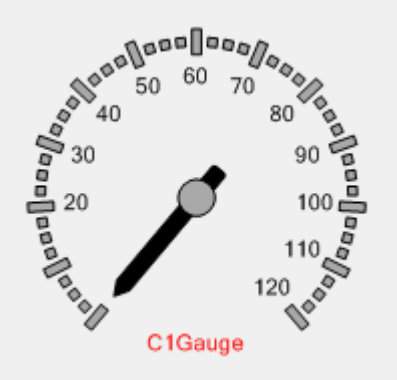
## What You've Accomplished

You've customized the appearance of the C1Gauge control. Next you will run the application.

# Step 3 of 3: Running the Application

In the previous steps you've created a new application, added a C1Gauge control to the form, and customized the control. All that's left is to run the application!

Select **Debug | Start Debugging** from the Visual Studio menu. The application will appear:




## What You've Accomplished

Congratulations! You have successfully created a C1Gauge control. There are many templates you can apply to your gauge. See Gauges for WinForms Appearance for more information.

# Gauges for WinForms Components

**Gauges for WinForms** represents information in the form of radial and linear gauges. It is also possible to adjust a gauge so it will work as an input control like a knob, track bar, or scroll bar.

# C1Gauge Objects and Usage

The following topics discover the usage of **C1Gauge**'s classes and components.

# C1Gauge Control

The C1Gauge control is a container control for gauges and shapes. The C1Gauge control supports both linear and radial gauges. Gauges display data, and can be customized using pointers and decorators, such as labels, tick marks, and ranges. Radial gauges can be circular, half-circular, clamshell, helical, or arc curve. Linear gauges can be horizontal, vertical, tilted, or free-formed.

The C1Gauge control also supports shapes. Shapes consist of static figures, captions, or images. Shapes include the following geometric figures: ellipse, rectangle, segment, and sector, each with a number of customizable settings. Shapes can also specify the clipping area for other elements, such as other shapes, decorators, and pointers.

A Pointer is a visual element that indicates the current value. Decorators can be bound to pointers. For example, you can bind a single label to the pointer so that the label will show the current value. This powerful technique is discussed in detail in the **Decorators** section.

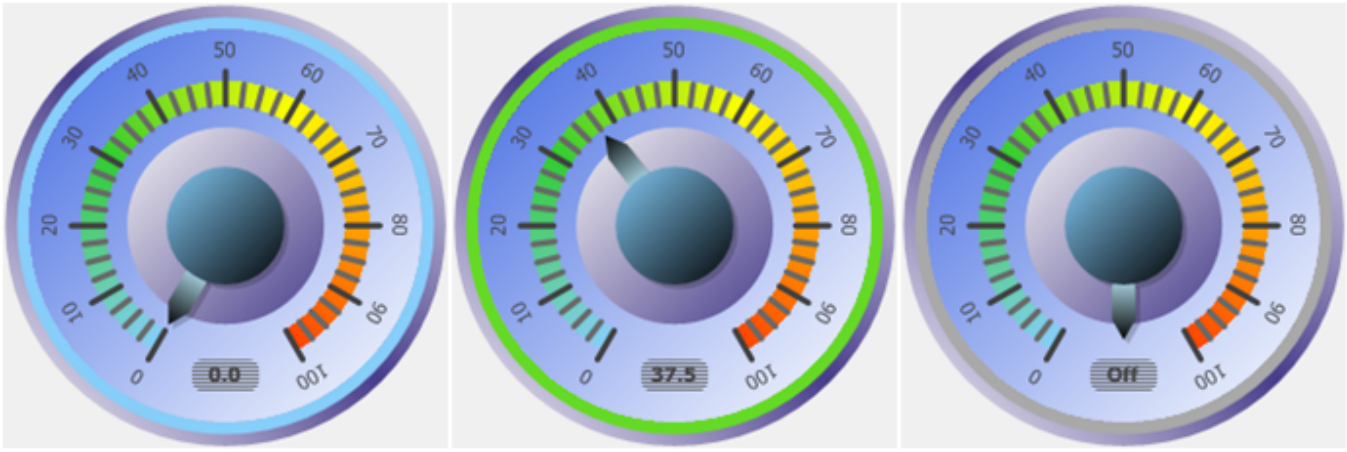
Except custom bitmaps, all graphics in **Gauges for WinForms** are vector-based. The C1Gauge container control and its gauges can be resized as you choose. You can specify the absolute or proportional positions for individual gauges and their aspect ratios.

# C1Gauge Composition

The C1Gauge control is made up of several layered elements when gauges are placed inside of the control. For example, the overall composition of a **C1Gauge** control with a single **C1RadialGauge** is the following (from the backmost to foremost layer):

- The control's **BackColor** and/or the **BackgroundImage** is the most background layer
- **C1Gauge.FaceShapes** is the next layer
- **C1RadialGauge.FaceShapes**
- **C1RadialGauge.Decorators**
- **C1RadialGauge.Pointers**
- **C1RadialGauge.Cap**
- **C1RadialGauge.CoverShapes**
- **C1Gauge.CoverShapes** is over everything.

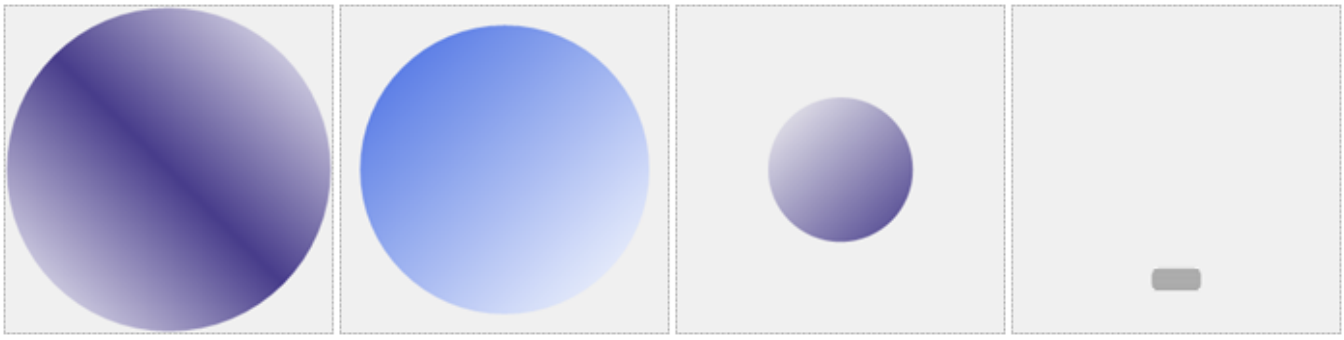
Let's take a look at the composition of a simple radial gauge (see the 'Interactive' sample in the GaugeDemo application for more details). The following images show a few states of the whole gauge.



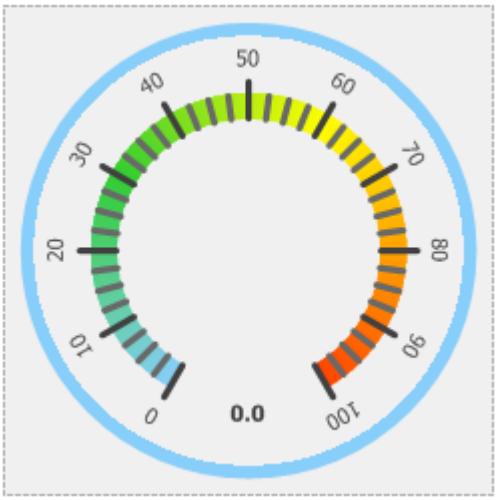

This gauge consists of the following parts: face shapes, decorators, pointer, and pointer cap.

Face shapes are the background of a gauge.

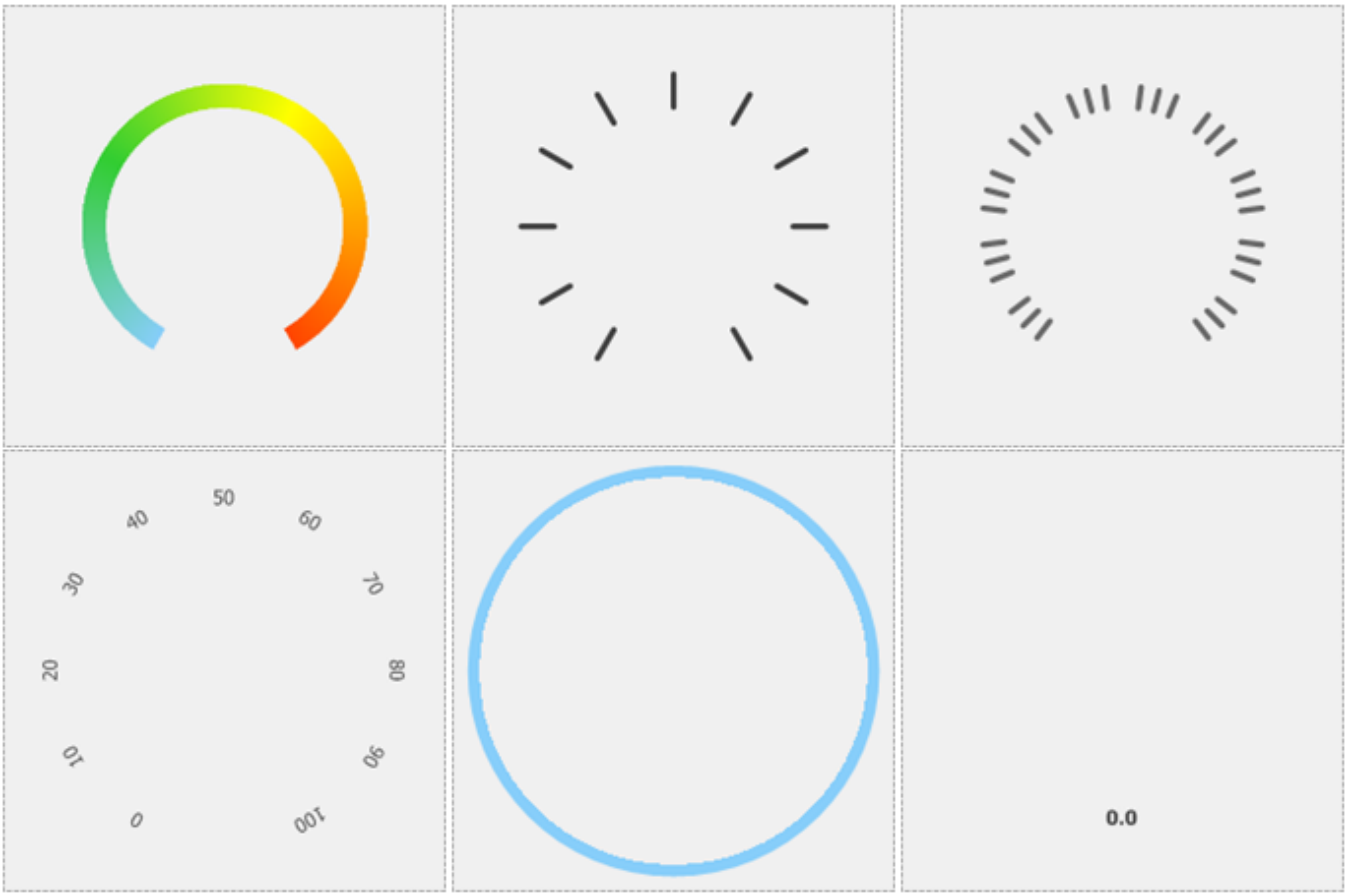
There are four shapes: three ellipses (**C1GaugeEllipse**) and a rectangle with rounded corners (**C1GaugeRectangle**).



Decorators show the scale ranges, markers, and labels.




This gauge contains a range (**C1GaugeRange**), major and minor markers (**C1GaugeMarks**), scale labels (**C1GaugeLabels**), the bound state mark (**C1GaugeSingleMark**), and a label that shows the current value (**C1GaugeSingleLabel**).




The topmost parts of this gauge are the pointer (**C1GaugePointer**) and the pointer cap (**C1GaugeCap**).

## C1Gauge Properties

The C1Gauge control includes several properties that allow you to customize the appearance and behavior of the control. The main properties of the C1Gauge control include:

- **C1Gauge.Gauges** – the collection of gauges that can appear in the container control.
- **C1Gauge.FaceShapes** – the collection of "background" shapes that appear between the control's background and gauges.
- **C1Gauge.CoverShapes** – these are the "foreground" shapes that appear on top of everything.
- **C1Gauge.Font** – specifies the default font for all gauge labels.
- **C1Gauge.ForeColor** – sets the default color for all borders and labels in gauges and shapes.
- **C1Gauge.BackColor** – sets the default filling color.
- **C1Gauge.Shadow** – the default settings for all gauge shadows.
- **C1Gauge.CacheBackground** and **C1Gauge.CacheForeground** – these properties specify whether all static background and/or foreground elements should be painted into off-screen images. Such images are drawn quickly when the current value changes. This improves performance at the cost of additional memory consumption.
- **C1Gauge.SupportsTransitionEffect** – this property enables the ability for smooth transition between visual states of the control when the user calls the **C1Gauge.EndUpdate()** method with the 'duration' parameter.
- **C1Gauge.FramesPerSecond** – sets the maximum frequency of repainting of the gauge container. You may decrease the value of this property to improve overall performance.
- **C1Gauge.Selectable** – if True, the control is selectable and can receive input focus.

When some gauge is selected on the designer surface, it's easy to move selection to the owner **C1Gauge** control by clicking in the grab handle that appears at the top left corner of the control. An alternative way would be pressing the **Escape** key – this moves the selection to the owner control as well.

## Gauge Components

Available gauges include the C1RadialGauge and C1LinearGauge components (both are derived from the C1GaugeBase class). Gauges aren't Controls; they are simple Components that can be added to the **C1Gauge**.**Gauges** collection.

You can select gauges on the designer surface and edit their properties/events in the standard property grid, in the special **Item Editor**, or in the collection editor for the **C1Gauge**.**Gauges** collection property. Gauge's designer supports the standard operations, such as Cut, Copy, Paste, Delete, "Bring to Front", "Send to Back". Also, you can right-click individual gauges to display their context menu. This allows quick editing of the **C1GaugeBase.FaceShapes**/**C1GaugeBase.CoverShapes**, and **C1GaugeBase.Decorators** collections.

Gauges can be saved as XML templates, and then loaded from XML files. You can also save/load the layout and appearance settings for individual gauges.

You can bind gauge to a data source using the standard .NET data-binding technique. If there are several pointers in the gauge you can bind these pointers to different data sources (up to 5 pointers for a gauge).

The main properties of the base **C1GaugeBase** class include:

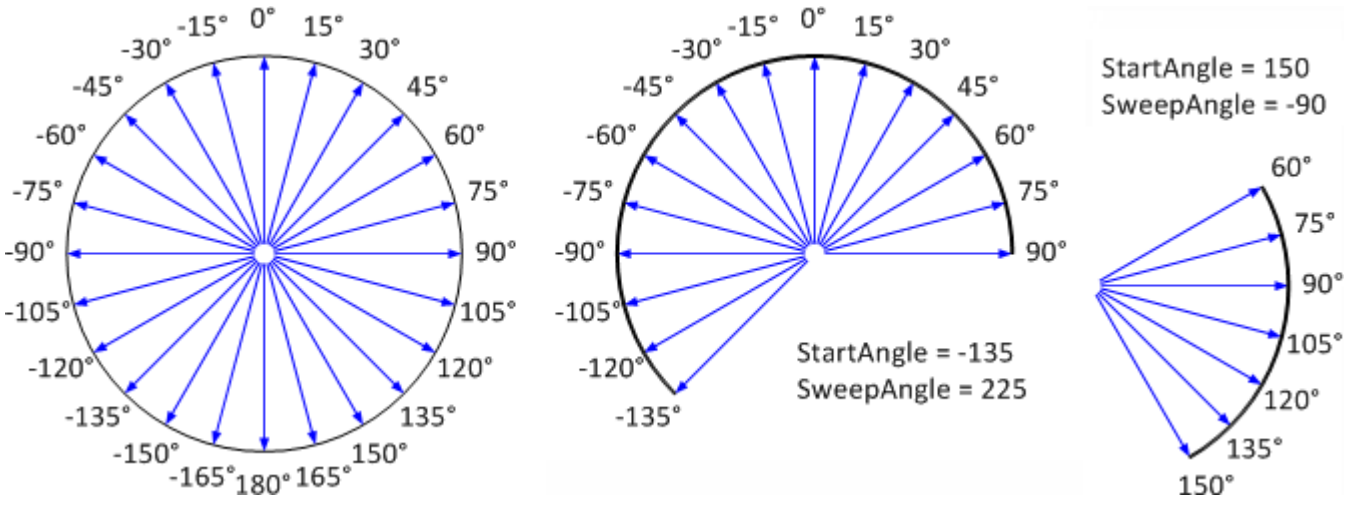
- **C1GaugeBase.Decorators** – the collection of various decorators, such as labels and tick marks.
- **C1GaugeBase.FaceShapes** and **C1GaugeBase.CoverShapes** – two collections of shapes that appear behind everything (**FaceShapes**) and above everything (**CoverShapes**) in the gauge.
- **C1GaugeBase.Pointer** – this is the main pointer of the gauge. You can hide the main pointer but can't delete it.
- **C1GaugeBase.MorePointers** – the collection of other pointers.
- **C1GaugeBase.Value** – the current value of the main pointer.
- **C1GaugeBase.Minimum** and **C1GaugeBase.Maximum** – specify the lower and upper bounds for all pointer values, for example scaled values.
- **C1GaugeBase.Viewport** – specifies the bounds of the gauge working area.

In the following topics, you'll see the specific properties of radial and linear gauges.

## C1RadialGauge Component

C1RadialGauge has the center point, radius, start and sweep angles. The PointerOriginX and PointerOriginY properties specify the center of the polar coordinate system associated with the gauge. The height or width of the working area (which is lesser) becomes the base dimension. The Radius property specifies a portion of the base dimension whose length is 100 in logical coordinates.

The StartAngle and SweepAngle properties provide the possible range of values for the angular coordinate. As opposed to the standard polar coordinates, the angle of 0° corresponds to the direction from center upwards.



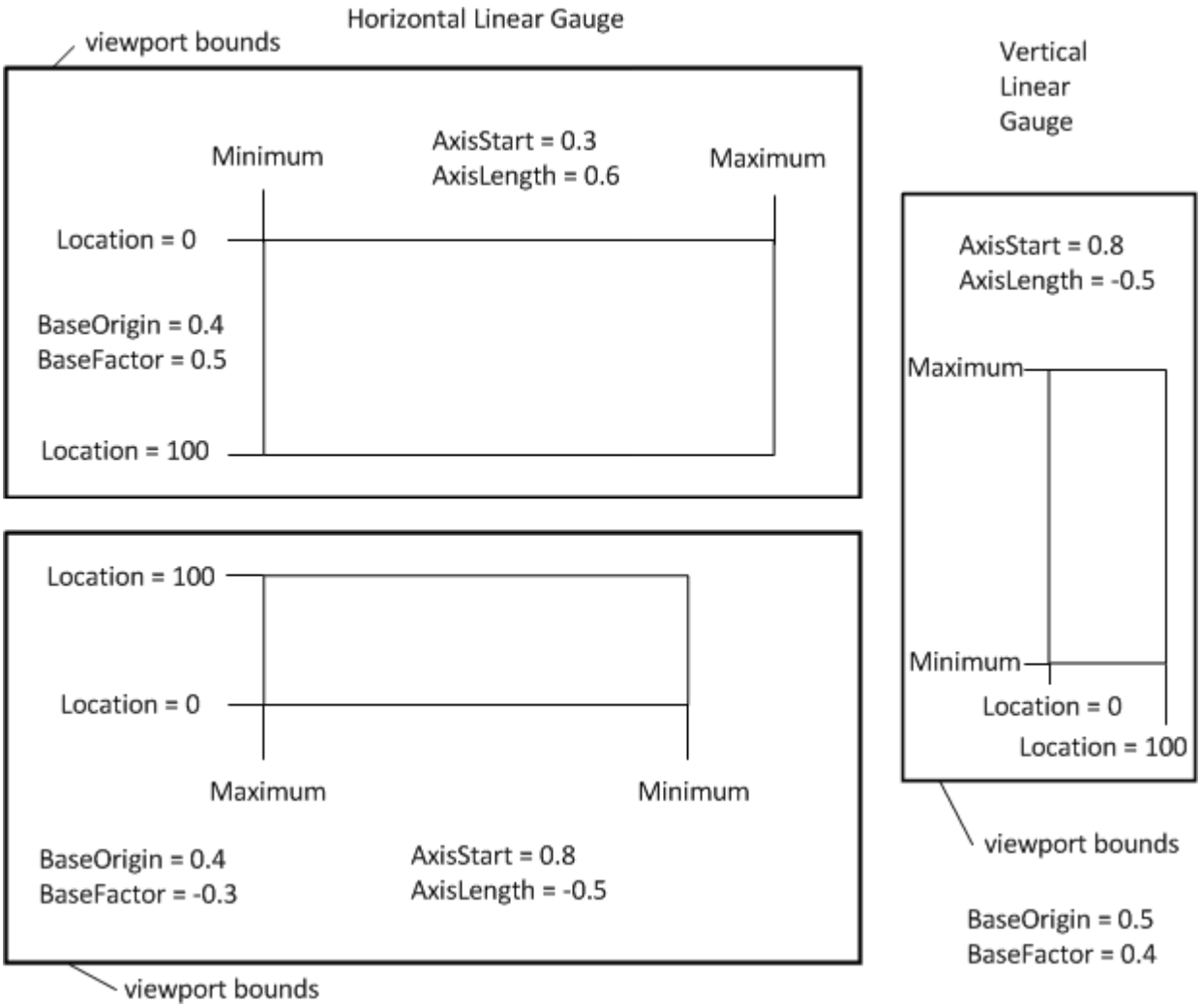

The **StartAngle** property defines an angle for the **C1GaugeBase.Minimum** value. The (**StartAngle** + **SweepAngle**) angle corresponds to the **C1GaugeBase.Maximum** value. You can reverse the direction of angular coordinate by setting the IsReversed property to **True**.

## C1LinearGauge Component

The horizontal linear gauge uses height of the working area as the base dimension. The vertical gauge uses area's width as the base dimension. C1LinearGauge has the BaseOrigin property that specifies a position (as a portion of the

base dimension) where the transversal axis starts. The BaseFactor property sets a portion of the base dimension whose length is 100 in logical coordinates.

The AxisStart and AxisLength properties specify the start and length of the longitudinal (value) axis. If the IsReversed property is set to **True** the direction of the longitudinal axis becomes opposite to the direction where the values increase. The next image shows a few available options for the linear gauge coordinate system.



# Design-Time Support

**Gauges for WinForms** provides visual editing to make it easier to create a schedule application. The following sections describe how to use **Gauges for WinForms**' design-time environment to configure the **Gauges for WinForms** controls:

## New Gauge Gallery

When you first add a **C1Gauge** control to the form in Design view, the **New Gauge Gallery** dialog box will appear. For more information, see the New Gauge Gallery topic.

## Smart Tags and Tasks Menus

You can invoke each control's tasks menu by clicking on the smart tag (▶) in the upper-right corner of the control. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each control. For more information on how to use the tasks menu in **Gauges for WinForms**, see C1Gauge Tasks Menu.

## Context Menus

You can invoke each control's context menu by right-clicking the component or control in Design view. The context menu includes common actions when using the control. For more information on how to use the context menu in **Gauges for WinForms**, see C1Gauge Context Menu or C1LinearGauge and C1RadialGauge Context Menus.

## Designers

You can easily configure the **Gauges for WinForms** components at design time by using the associated collection editors. For more information on the **Gauges for WinForms** designers, see the Gauges for WinForms Collection Editors topic.

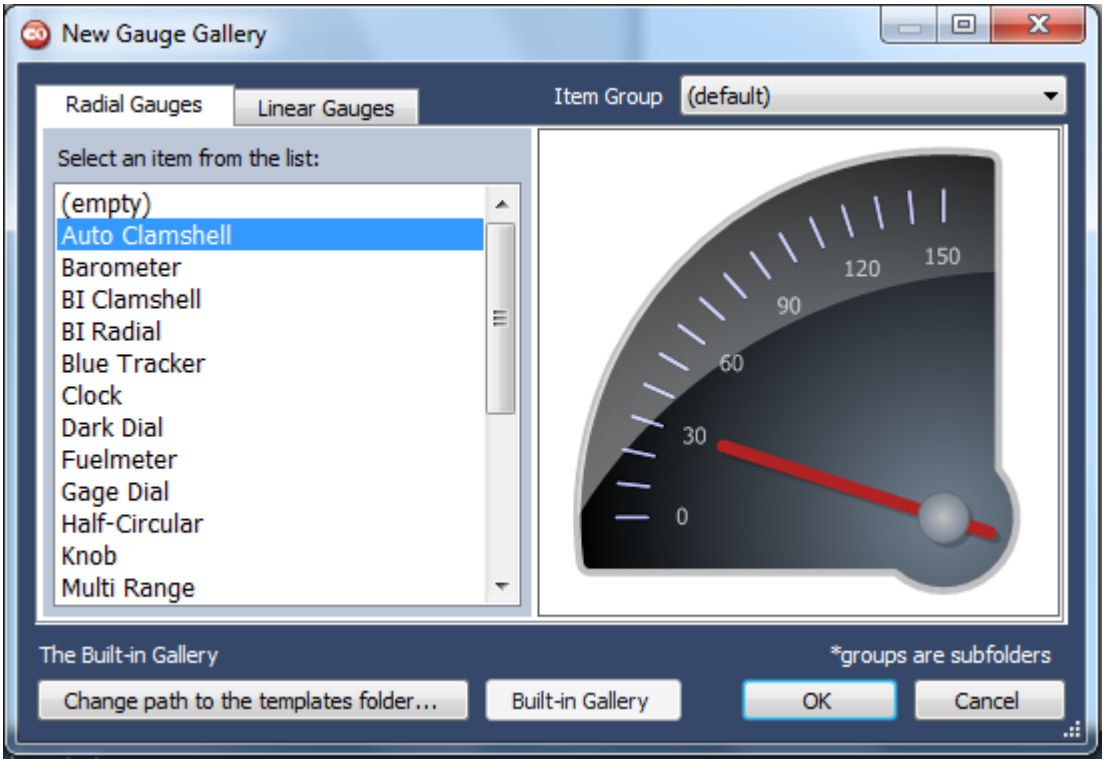
## Properties Window

You can also easily configure **Gauges for WinForms** at design time using the Properties window in Visual Studio. You can access the Properties window by right-clicking the control and selecting **Properties**.

# New Gauge Gallery

Select the C1Gauge control in your project and click the smart tag. Select **Add New Gauge** from the **C1Gauge Tasks** menu. The **New Gauge Gallery** dialog box appears, allowing you to pick and choose templates and template groups.

You can also access the **New Gauge Gallery** dialog box by selecting **Add New Gauge** from the C1Gauge Context Menu. The **New Gauge Gallery** dialog box appears similar to the following:




In this dialog box you can choose the initial appearance of your gauge from default or existing custom templates. The **New Gauge Gallery** dialog box operates as follows:

- **Radial Gauges tab**

  Select the **Radial Gauges** tab to select from available radial gauges. You can also choose the **(empty)** option to add an unformatted **C1RadialGauge** item.

- **Linear Gauges**

  Select the **Linear Gauges** tab to select from available horizontal and vertical linear gauges. You can also choose the **(empty)** option to add an unformatted **C1LinearGauge** item.

- **Item Group drop-down box**

  Open the **Item Group** drop-down box to choose a built-in template group. Choices include Advanced, OfficeBlack, OfficeBlue, OfficeSilver, and Windows 7. For more information, see the Gauges for WinForms Appearance topics.

- **Change path to the templates folder button**

  Click the **Change path to the templates folder** button to navigate to a folder that contains custom template files. Once you choose a folder, templates in that folder will appear in the dialog box. You can return to the built-in templates by clicking the **Built-in Gallery** button.

- **Built-in Gallery button**

  Click the **Built-in Gallery** button to return to the built-in template options. This button is only selectable when you are currently viewing custom templates.

- **OK button**

  Click the **OK** button to save any changes or selections you have made and close the **New Gauge Gallery** dialog box. The gauge you added will then appear in the **C1Gauge** control.

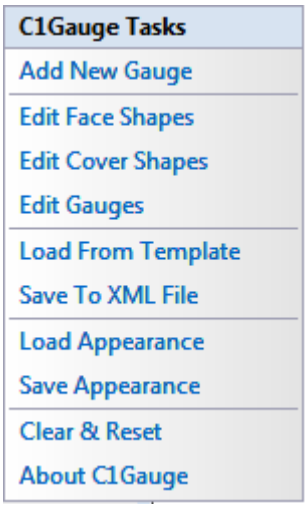
- **Cancel button**

  Click the **Cancel** button to cancel any changes or selections you have made and close the **New Gauge Gallery** dialog box.

## C1Gauge Tasks Menu

In the **C1Gauge Tasks** menu you can quickly and easily add, edit, load, and save gauges.

To access the **C1Gauge Tasks** menu, click on the smart tag (▶) in the upper right corner of the control. This will open the **C1Gauge Tasks** menu which appears like the following:




The **C1Gauge Tasks** menu operates as follows:

- **Add New Gauge**

  Selecting the **Add New Gauge** option opens the **New Gauge Gallery** when you can choose a new gauge to add to the **C1Gauge** container control. You can also choose a template and set some options for the new gauge. For more information, see the New Gauge Gallery topic.

- **Edit Face Shapes**

  Selecting the **Edit Face Shapes** option opens the **C1Gauge.FaceShapes Collection Editor** where you can add and remove items in the **FaceShapes** collection, and customize properties on each item in the collection. For more information, see the FaceShapes Collection Editor topic.

- **Edit Cover Shapes**

  Selecting the **Edit Cover Shapes** option opens the **C1Gauge.CoverShapes Collection Editor** where you can add and remove items in the **CoverShapes** collection, and customize properties on each item in the collection. For more information, see the CoverShapes Collection Editor topic.

- **Edit Gauges**

  Selecting the **Edit Gauges** option opens the **C1Gauge.Gauges Collection Editor** where you can add and remove members in the **Gauges** collection, and customize properties on each gauge in the collection. For more information, see the Gauges Collection Editor topic.

- **Load From Template**

Selecting the **Load From Template** option opens the **Load C1Gauge From Template** dialog box where you can choose a new built-in or custom template to customize the appearance of the gauge. See Loading a C1Gauge from a Template for more information.

- **Save To XML File**

  Selecting the **Save to XML File** option opens the **Save C1Gauge To XML File** dialog box where you can select a location to save the gauge template. For more information, see the Saving a C1Gauge to an XML File topic.

- **Load Appearance**

  Selecting the **Load Appearance** option opens the **Load C1Gauge View From XML File** dialog box where you can select an XML file to load. See Loading a C1Gauge View from an XML File for more information.

- **Save Appearance**

  Selecting the **Save Appearance** option opens the **Save C1Gauge View To XML File** dialog box where you can save the current gauge's appearance as an XML file. For more information, see the Saving a C1Gauge View to an XML File topic.

- **Clear & Reset**

  Clicking **Clear & Reset** clears the C1Gauge control and resets its properties to the default values.

- **About C1Gauge**

  Selecting the **About C1Gauge** option opens the **Save About C1Gauge** dialog box which is helpful in finding the build number of the control.

## C1Gauge Context Menu

In the **C1Gauge** context menu you can quickly and easily add, edit, load, and save gauges.

To access the **C1Gauge** context menu, right-click the C1Gauge control. The **C1Gauge** context menu operates as follows:

- **Launch Item Editor**

  Clicking **Launch Item Editor** opens the selected item's properties dialog box where you can specify properties for the control, decorator, pointer, and so on.

- **Add New Gauge**

  Selecting the **Add New Gauge** option opens the **New Gauge Gallery** when you can choose a new gauge to add to the **C1Gauge** container control. You can also choose a template and set some options for the new gauge. For more information, see the New Gauge Gallery topic.

- **Edit Face Shapes**

  Selecting the **Edit Face Shapes** option opens the **C1Gauge.FaceShapes Collection Editor** where you can add and remove items in the **FaceShapes** collection, and customize properties on each item in the collection. For more information, see the FaceShapes Collection Editor topic.

- **Edit Cover Shapes**

  Selecting the **Edit Cover Shapes** option opens the **C1Gauge.CoverShapes Collection Editor** where you can add and remove items in the **CoverShapes** collection, and customize properties on each item in the collection. For more information, see the CoverShapes Collection Editor topic.

- **Edit Gauges**

  Selecting the **Edit Gauges** option opens the **C1Gauge.Gauges Collection Editor** where you can add and remove members in the **Gauges** collection, and customize properties on each gauge in the collection. For more information, see the Gauges Collection Editor topic.

- **Load From Template**

  Selecting the **Load From Template** option opens the **Load C1Gauge From Template** dialog box where you can choose a new built-in or custom template to customize the appearance of the gauge.

- **Save To XML File**

  Selecting the **Save to XML File** option opens the **Save C1Gauge To XML File** dialog box where you can select a location to save the gauge template.

- **Load Appearance**

  Selecting the **Load Appearance** option opens the **Load C1Gauge View From XML File** dialog box where you can select an XML file to load.

- **Save Appearance**

  Selecting the **Save Appearance** option opens the **Save C1Gauge View To XML File** dialog box where you can save the current gauge's appearance as an XML file.

- **Clear & Reset**

  Clicking **Clear & Reset** resets the contents of the C1Gauge control.

## C1LinearGauge and C1RadialGauge Context Menus

In the C1LinearGauge and C1RadialGauge context menus you can quickly and easily edit, load, and save gauges.

To access the **C1LinearGauge** and **C1RadialGauge** context menu, right-click the **C1LinearGauge** or **C1RadialGauge** component. The **C1LinearGauge** and **C1RadialGauge** context menus operate as follows:

- **Launch Item Editor**

  Clicking **Launch Item Editor** opens the selected item's properties dialog box where you can specify properties for the control, decorator, pointer, and so on.

- **Edit Face Shapes**

  Selecting the **Edit Face Shapes** option opens the **C1Gauge.FaceShapes Collection Editor** where you can add and remove items in the **FaceShapes** collection, and customize properties on each item in the collection. For more information, see the FaceShapes Collection Editor topic.

- **Edit Cover Shapes**

  Selecting the **Edit Cover Shapes** option opens the **C1Gauge.CoverShapes Collection Editor** where you can add and remove items in the **CoverShapes** collection, and customize properties on each item in the collection. For more information, see the CoverShapes Collection Editor topic.

- **Edit Decorators**

  Selecting the **Edit Decorators** option opens the **C1RadialGauge.Decorators Collection Editor** or **C1LinearGauge.Decorators Collection Editor** where you can add and remove labels, ranges, and marks in the

**Decorators** collection, and customize properties on each item in the collection. For more information, see the Decorators Collection Editor topic.

- **Edit Pointers**

  Selecting the **Edit Pointers** option opens **MorePointers Collection Editor** where you can add, remove, and edit pointers in the **MorePointers** collection. For more information, see the MorePointers Collection Editor topic.

- **Load From Template**

  Selecting the **Load From Template** option opens the **Load C1Gauge From Template** dialog box where you can choose a new built-in or custom template to customize the appearance of the gauge.

- **Save To XML File**

  Selecting the **Save to XML File** option opens the **Save C1Gauge To XML File** dialog box where you can select a location to save the gauge template.

- **Load Appearance**

  Selecting the **Load Appearance** option opens the **Load C1Gauge View From XML File** dialog box where you can select an XML file to load.

- **Save Appearance**

  Selecting the **Save Appearance** option opens the **Save C1Gauge View To XML File** dialog box where you can save the current gauge's appearance as an XML file.

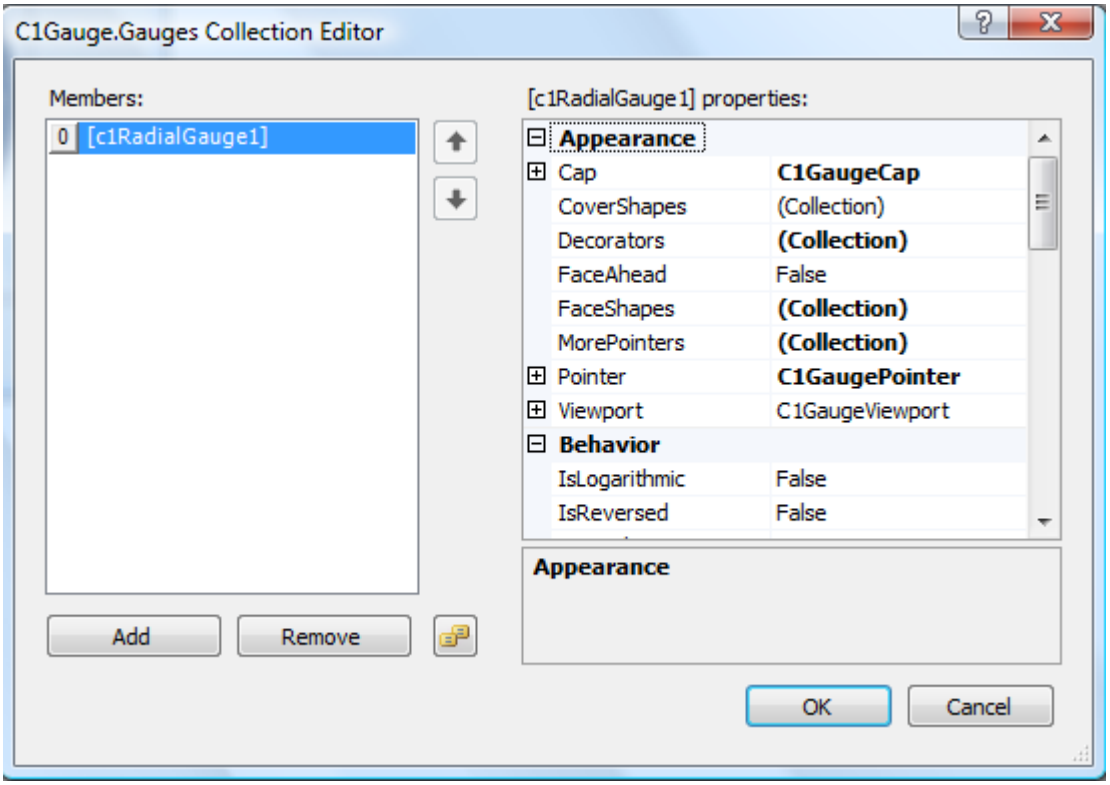
## Gauges for WinForms Collection Editors

**Gauges for WinForms** includes several collection editors that make customizing gauges easier. The following topics detail how to access and use these editors.

## Gauges Collection Editor

The **C1Gauge.Gauges Collection Editor** allows you to add, remove, and edit gauges in the Gauges collection. You can access the **Gauges Collection Editor** by selecting the **Edit Gauges** option from the C1Gauge Tasks menu or from the C1Gauge context menu or by clicking the **ellipses** button next to the **Gauges** item in **C1Gauge**'s Properties window.

Once open, the **C1Gauge.Gauges Collection Editor** will appear similar to the following image:



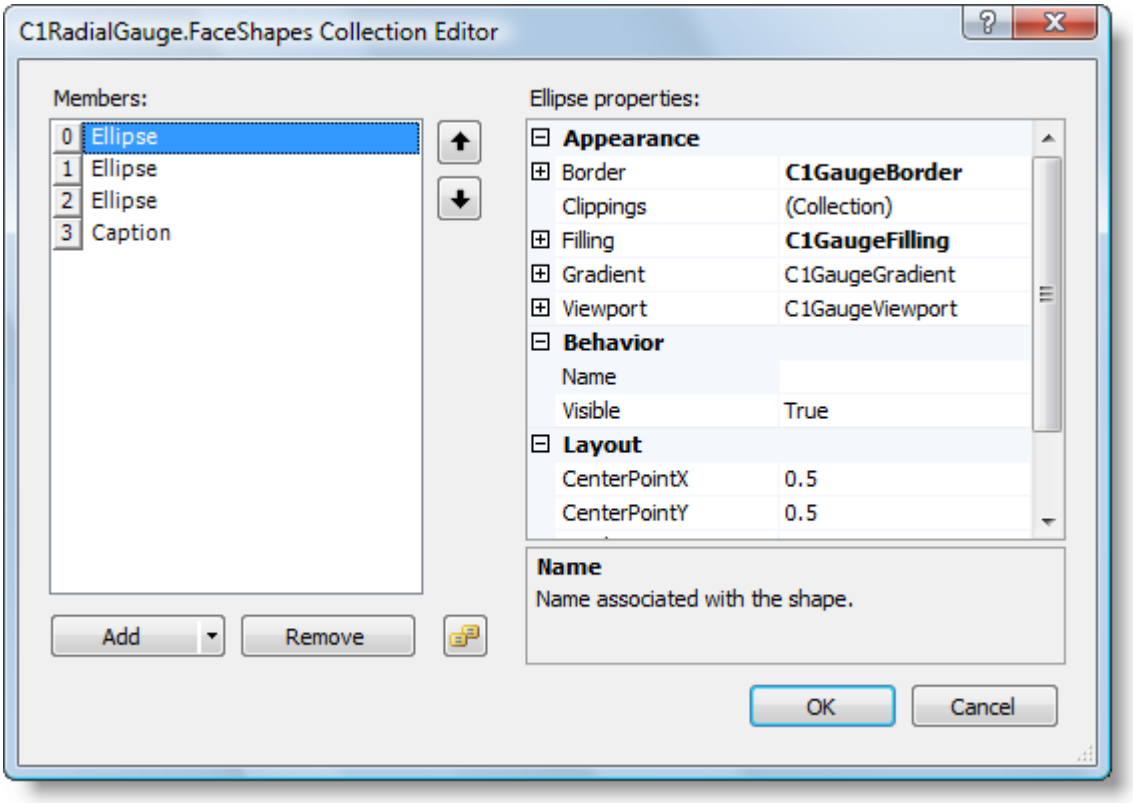

The left side of the editor lists C1RadialGauge and C1LinearGauge items hosted within the C1Gauge control. You can click the **Add** button to add a new gauge, which will open the **New Gauge Gallery** dialog box. To remove a gauge, select the gauge to remove and click the **Remove** button.

To edit properties on a gauge, pick an item in the **Members** list and edit its properties in the properties grid. To save your changes click the **OK** button. To close the dialog box without saving any changes, click the **Cancel** button. You can also double-click the C1Gauge item on the form to open the **Item Editor** and edit the properties.

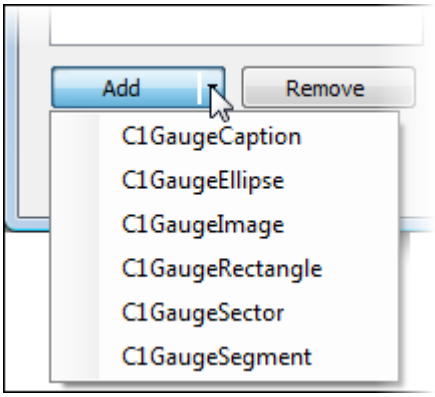
## FaceShapes Collection Editor

The **C1Gauge.FaceShapes Collection Editor**, **C1LinearGauge.FaceShapes Collection Editor**, and **C1RadialGauge.FaceShapes Collection Editor** allow you to add, remove, and edit items in the FaceShapes collection. You can access the **FaceShapes Collection Editor** by selecting the **Edit Face Shapes** option from the C1Gauge Tasks menu or from the C1Gauge context menu or C1LinearGauge and C1RadialGauge context menus or by clicking the **ellipses** button next to the **FaceShapes** item in C1Gauge, C1LinearGauge, or C1RadialGauges Properties window.

Once open, the **FaceShapes Collection Editor** will appear similar to the following image:




The left side of the editor lists shapes included in the collection. You can click the **Add** button to add a new caption, ellipse, image, rectangle, sector or segment:



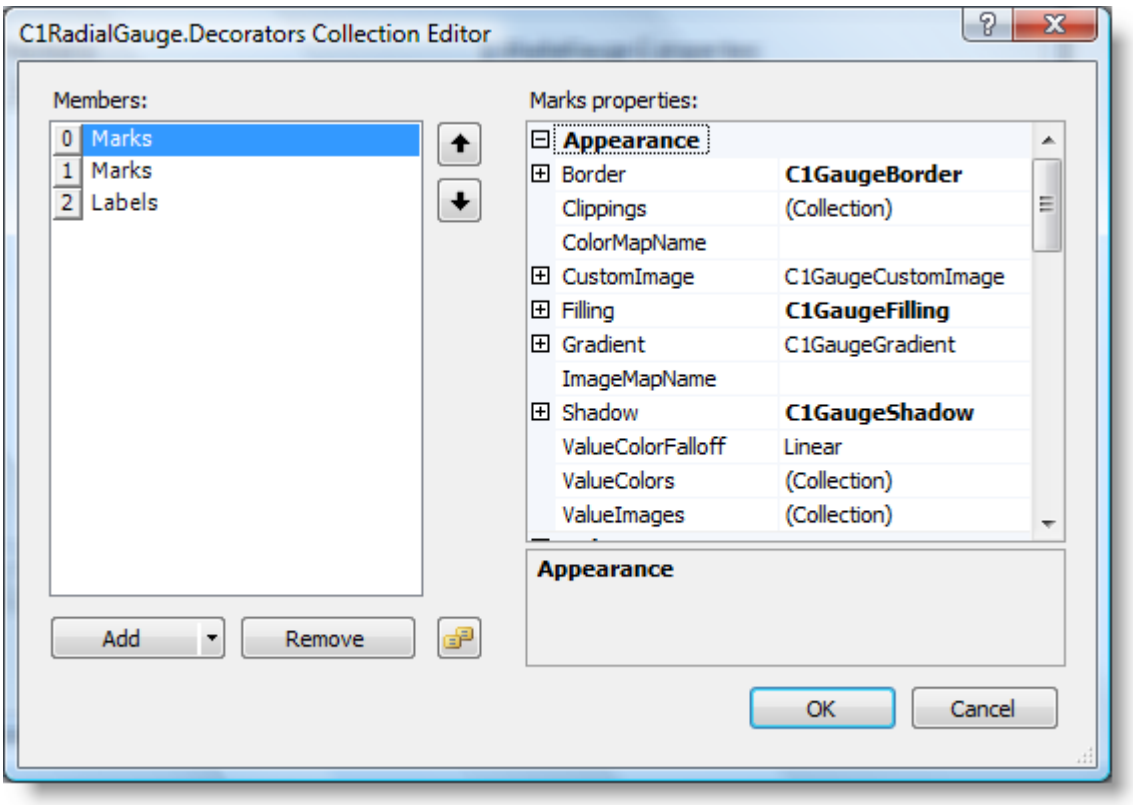

To remove an item, select the item to remove and click the **Remove** button. You can also change the order that shapes appear in by using the arrow buttons.

To edit properties on a shape, pick an item in the **Members** list and edit its properties in the properties grid. To save your changes click the **OK** button. To close the dialog box without saving any changes, click the **Cancel** button. You can also double-click the FaceShapes item on the form to open the **Item Editor** and edit the properties.

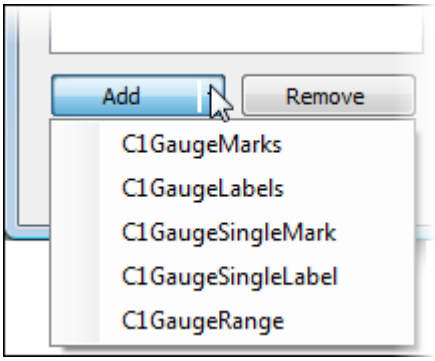
## Decorators Collection Editor

The **C1LinearGauge.Decorators Collection Editor** and **C1RadialGauge.Decorators Collection Editor** allow you to add, remove, and edit items in the Decorators collection. You can access the **Decorators Collection Editor** by selecting the **Edit Decorators** option from the C1LinearGauge and C1RadialGauge context menus or by clicking the **ellipses** button next to the **Decorators** item in **C1LinearGauge** or **C1RadialGauge**'s Properties window.

Once open, the **Decorators Collection Editor** will appear similar to the following image:




The left side of the editor lists the decorators included in the collection. You can click the **Add** button to add new marks, labels, a single mark, a single label, or a range:




To remove an item, select the item to remove and click the **Remove** button. You can also change the order that decorators appear in by using the arrow buttons.

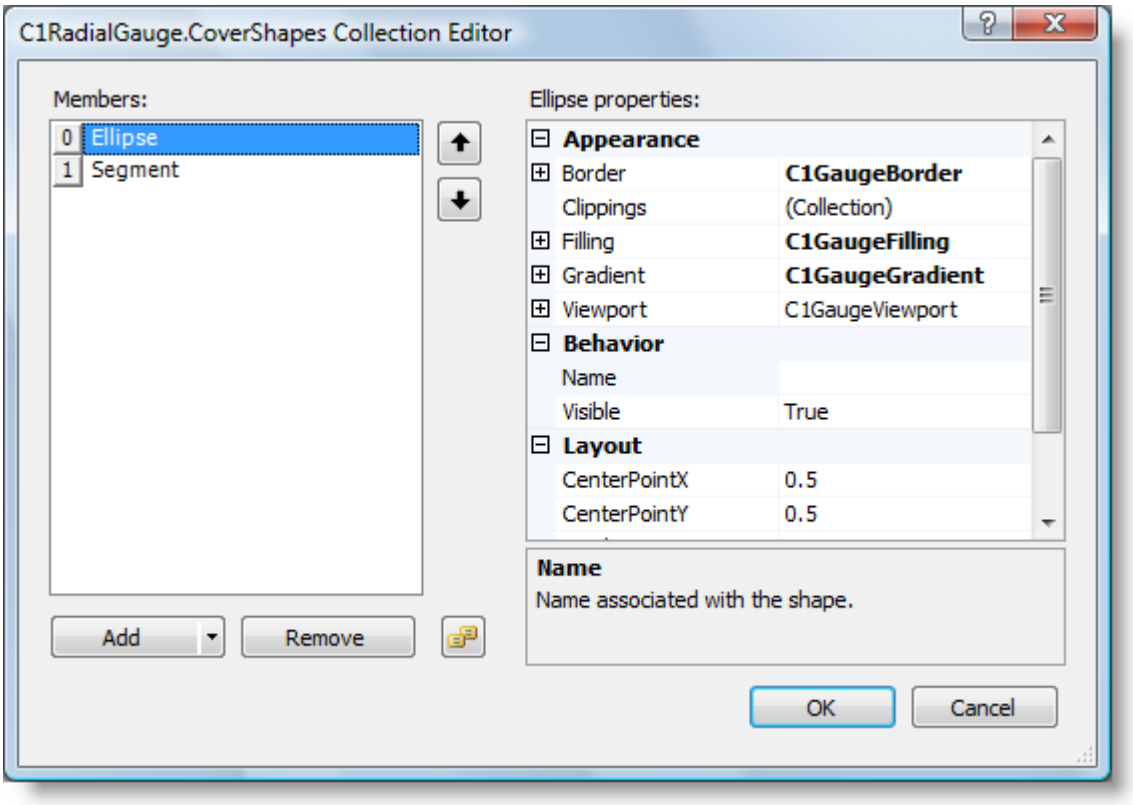
To edit properties on a decorator, pick an item in the **Members** list and edit its properties in the properties grid. To save your changes click the **OK** button. To close the dialog box without saving any changes, click the **Cancel** button. You can also double-click the Decorators item on the form to open the **Item Editor** and edit the properties.

## CoverShapes Collection Editor

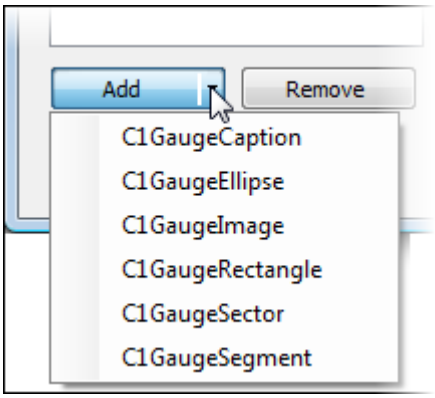
The **C1Gauge.CoverShapes Collection Editor**, **C1LinearGauge.CoverShapes Collection Editor**, and **C1RadialGauge.CoverShapes Collection Editor** allow you to add, remove, and edit items in the CoverShapes collection. You can access the **CoverShapes Collection Editor** by selecting the **Edit Cover Shapes** option from the

C1Gauge Tasks menu or from the C1Gauge context menu or C1LinearGauge and C1RadialGauge context menus or by clicking the **ellipses** button next to the **CoverShapes** item in **C1Gauge**, **C1LinearGauge**, or **C1RadialGauge**'s Properties window.

Once open, the **CoverShapes Collection Editor** will appear similar to the following image:




The left side of the editor lists shapes included in the collection. You can click the **Add** button to add a new caption, ellipse, image, rectangle, sector or segment:



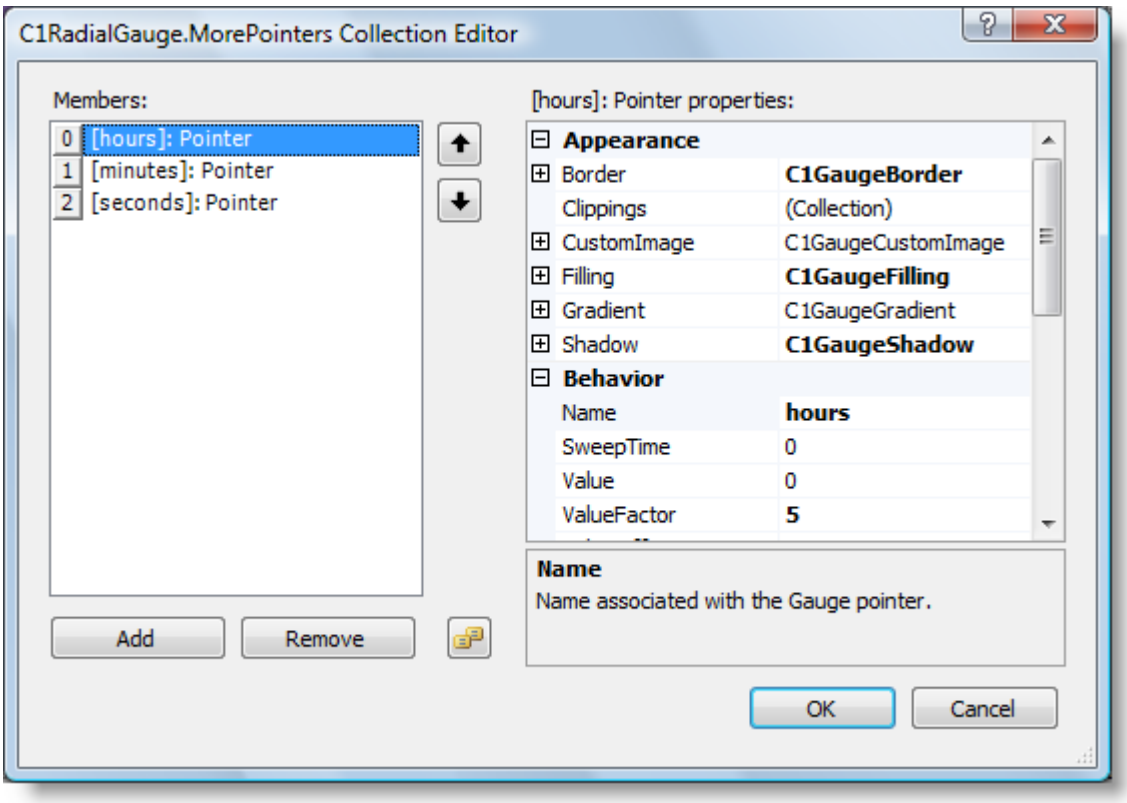

To remove an item, select the item to remove and click the **Remove** button. You can also change the order that shapes appear in by using the arrow buttons.

To edit properties on a shape, pick an item in the **Members** list and edit its properties in the properties grid. To save your changes click the **OK** button. To close the dialog box without saving any changes, click the **Cancel** button. You can also double-click the CoverShapes item on the form to open the **Item Editor** and edit the properties.

## MorePointers Collection Editor

The **C1LinearGauge.MorePointers Collection Editor** and **C1RadialGauge.MorePointers Collection Editor** allow you to add, remove, and edit items in the MorePointers collection. You can access the **MorePointers Collection Editor** by selecting the **Edit Pointers** option from the C1LinearGauge and C1RadialGauge context menus or by clicking the **ellipses** button next to the **MorePointers** item in **C1LinearGauge** or **C1RadialGauge**'s Properties window.

Once open, the **MorePointers Collection Editor** will appear similar to the following image:



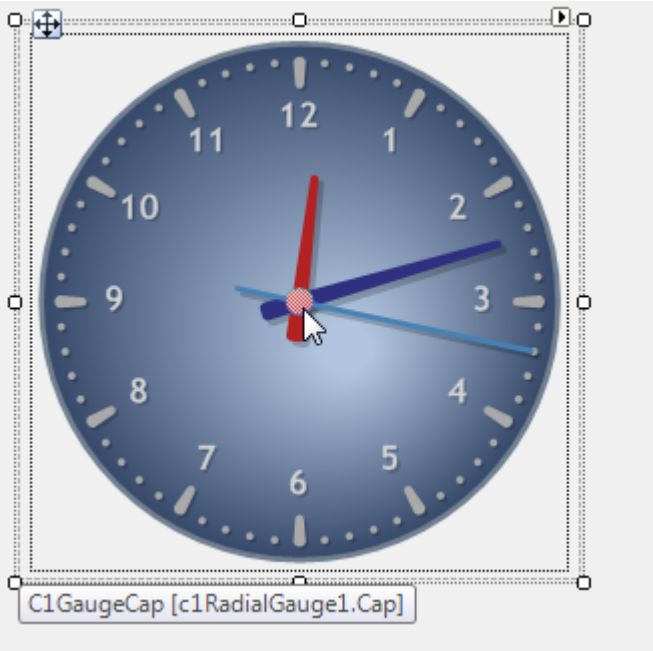

The left side of the editor lists the pointers included in the collection. You can click the **Add** button to add a new pointer. To remove a pointer, select the item to remove and click the **Remove** button. You can also change the order that pointers appear in by using the arrow buttons.

To edit properties on a pointer, pick an item in the **Members** list and edit its properties in the properties grid. To save your changes click the **OK** button. To close the dialog box without saving any changes, click the **Cancel** button. You can also double-click the MorePointers item on the form to open the **Item Editor** and edit the properties.

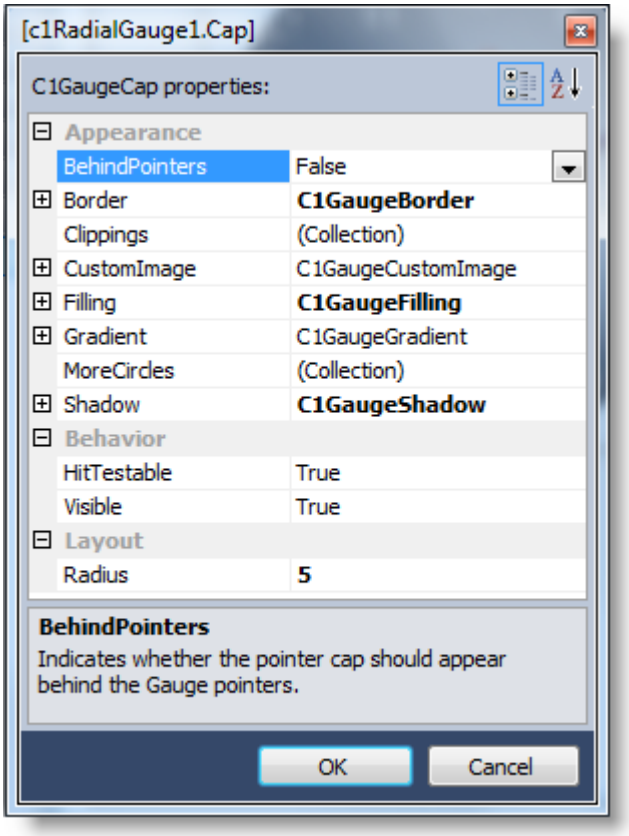
## Accessing Item Properties

You can easily access properties for any hit-testable **Gauge for WinForms** item at design time by double-clicking the item on your form and opening the **Item Editor**. You can also open the **Item Editor** by right-clicking an item and selecting **Launch Item Editor** from the context menu.

For example, double-click the C1GaugeCap on a C1RadialGauge.




The **C1RadialGauge1.Cap Item Editor** opens, allowing you to set the properties for the cap and to access collection editors.



# Working with Gauges for WinForms

**Gauges for WinForms** supports linear and radial gauges to provide an intuitive and attractive way to display information graphically. The following topics explain the main aspects of **Gauges for WinForms**.

# Gauge Positioning and Arrangement

The origin of the coordinate system is at the top-left corner of the C1Gauge control. It is possible to align multiple gauges in the same container control. Gauges can be overlapped or placed side-by-side. The fine-turning settings give an ability to maintain the aspect ratio and relative position of individual gauges when resizing the container control.

The C1GaugeViewport class lies in the core of these abilities.

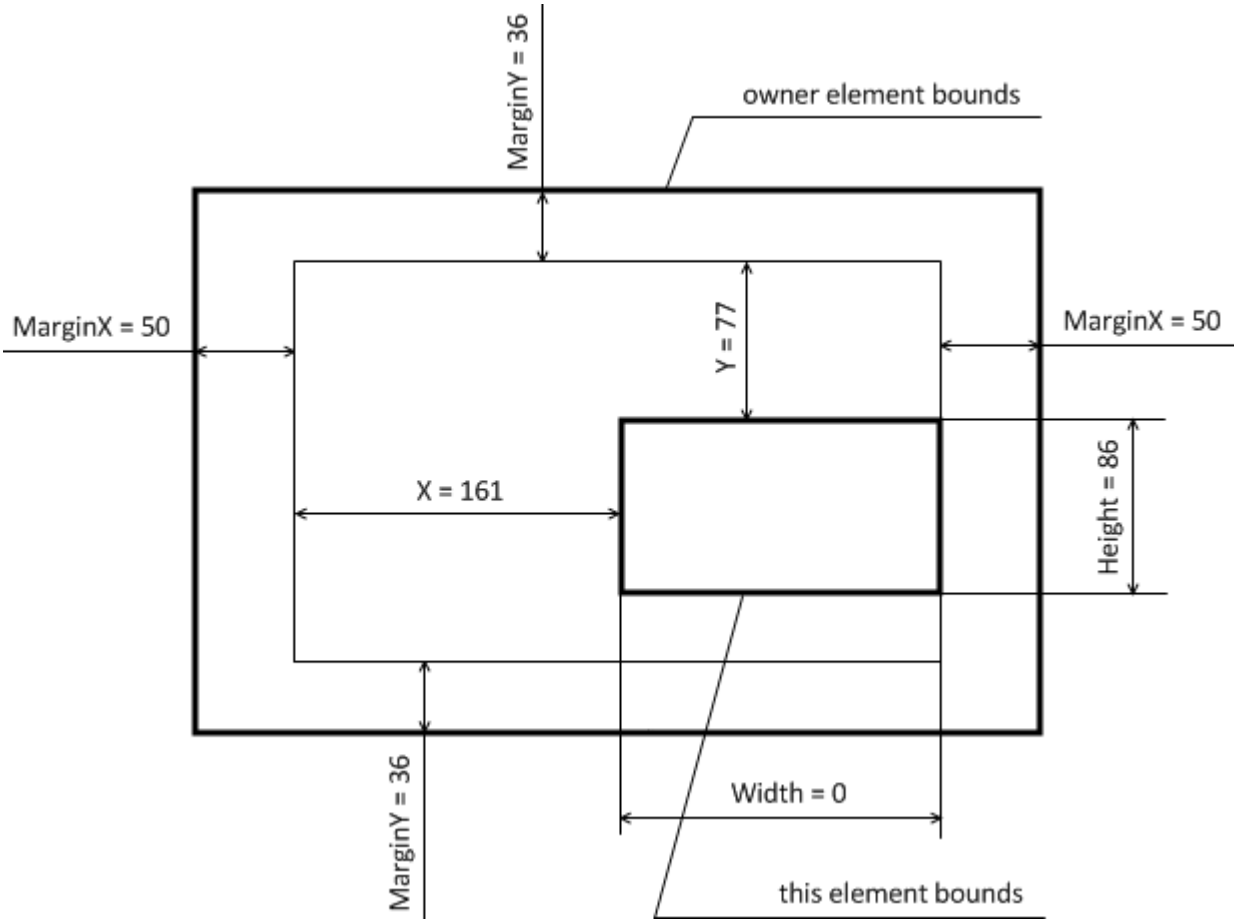
The following elements have the **Viewport** property: C1Gauge, C1GaugeBase, and C1GaugeBaseShape. They are related as parent-child-grandchild. So the client area of the container control becomes the basis for C1Gauge viewport. Viewport for an individual gauge is based on the owner C1Gauge viewport. Viewport for gauge shapes is count from the owner gauge's viewport.

Now let's consider how we can get the working area for a given element having its viewport settings and the owner element's working area.

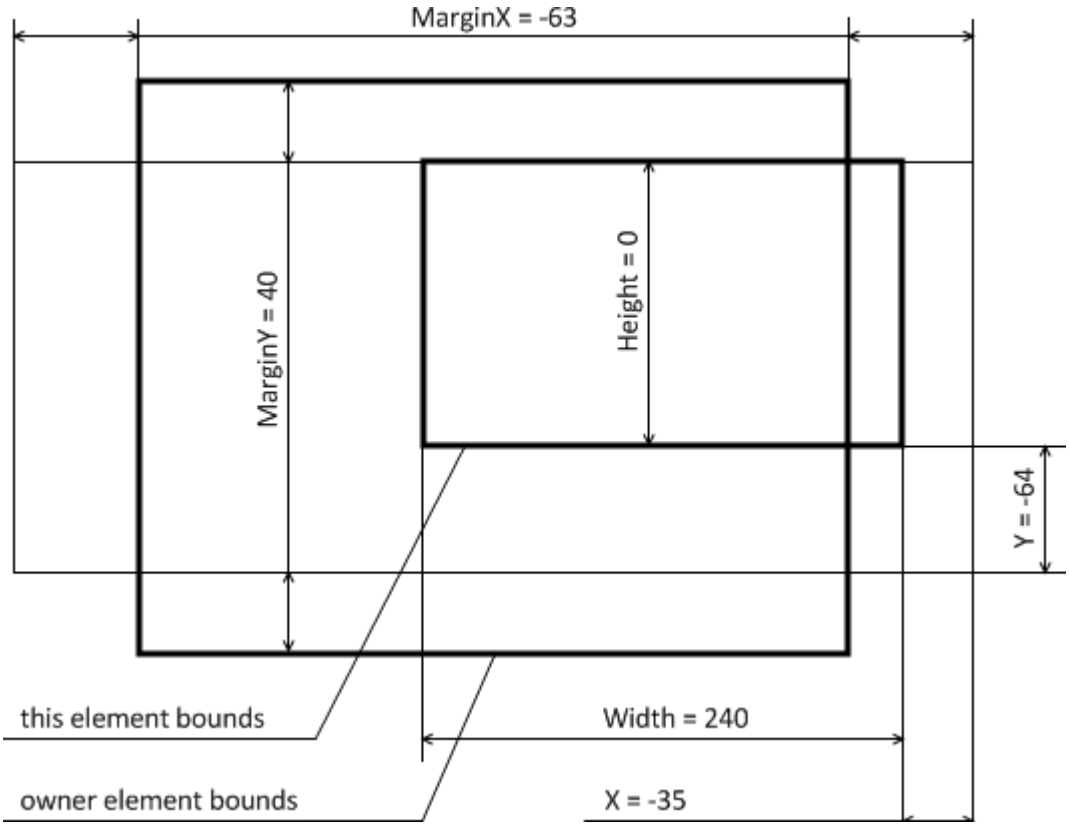
The following is list of the C1GaugeViewport object properties in the same order as they are applied:

- **C1GaugeViewport.TranslateX** and **C1GaugeViewport.TranslateY** – at the first stage we move the viewport by the specified relative amount along the X and Y axes. The amount of 1.0 corresponds to the whole width or height of the owner area.
- **C1GaugeViewport.ScaleX** and **C1GaugeViewport.ScaleY** – scale the width and height of the viewport by the specified relative amount.
- **C1GaugeViewport.MarginX** and **C1GaugeViewport.MarginY** – specify the horizontal and vertical margins, in pixels. If these margins are negative the working area becomes wider than the original bounds.
- **C1GaugeViewport.X** and **C1GaugeViewport.Y** – set the offset of the viewport, in pixels. Negative offset is count from the right or bottom edge of the owner area.
- **C1GaugeViewport.Width** and **C1GaugeViewport.Height** – specify the width and height of the viewport, in pixels. These properties equal to 0 by default. That extends the viewport till the opposite edge of the owner area. For example, if the **X** property value is negative the viewport will be extended till the left edge. Negative width/height extends the viewport backwards. For example, to move the working area to the right top corner you may set **X**=-100, **Width**=-100, **Y**=0, **Height**=0.
- **C1GaugeViewport.AspectRatio** – sets the fixed ratio of width to height of the viewport. Maintaining the aspect ratio may cause moving the working area.
- **C1GaugeViewport.AspectPinX** and **C1GaugeViewport.AspectPinY** – at the last stage we set the relative position that remains fixed when the working area moves in order to maintain the aspect ratio. For a radial gauge these properties are equal to **C1RadialGauge.PointerOriginX** and **C1RadialGauge.PointerOriginY** by default.

The next image shows a viewport that has all pixel-based properties set to positive values, except the **Width** which is equal to 0 (other properties have their default values):




Next sample shows a viewport where the **MarginX**, **X**, and **Y** properties are negative and **Height** is equal to 0.



## Gauge Pointers

Gauges may have one or several pointers. Pointer appears as a figure whole position depends on the current value. Each gauge object has the main pointer (**C1GaugeBase**.**Pointer**) and the collection of other pointers (**C1GaugeBase**.**MorePointers**).

Each pointer has the Value property. The lower and upper bounds for the pointer value is specified by the **C1GaugeBase.Minimum** and **C1GaugeBase.Maximum** property of the owner gauge. If it doesn't belong to the given interval you may use the **C1GaugePointer.ValueOffset** and **C1GaugePointer.ValueFactor** properties to coerce the source value, for example, change the unit of measure.

When the Value property changes the pointer redraws at the new location immediately, by default. The SweepTime property allows smooth moving the pointer. It sets the amount of time that is taken to move the pointer from Minimum to Maximum.

There are a few properties that affect pointer's appearance. You can choose among the predefined pointer shapes, create a custom shape, or display a custom image for the pointer. The **C1GaugeBase.Offset**, **C1GaugeBase.Length**, and **C1GaugeBase.Width** properties specify the offset, length, and width of the pointer in logical coordinates at the scale's minimum position. It is possible that the offset and/or length of the pointer will be changed linearly from minimum to maximum values. So there are also the **C1GaugeBase.Offset2** and **C1GaugeBase.Length2** properties that specify the offset and length at the scale's maximum value.

Pointer values can be bound to a data source. The **C1GaugeBase**.**DataSource** and **C1GaugeBase**.**DataMember** properties allow easy binding to the main pointer. There is also the special **C1GaugeBase**.**BoundValue** property. This is the same as Value (i.e., the value of the main pointer) but it is of type Object and returns DBNull.Value instead of Double.NaN if the pointer is "switched off". The next properties: **C1GaugeBase**.**MorePointersValue_0**, **C1GaugeBase**.**MorePointersValue_1**, **C1GaugeBase**.**MorePointersValue_2**, **C1GaugeBase**.**MorePointersValue_3** allow binding to the first four elements of the **C1GaugeBase**.**MorePointers** collection.

## C1RadialGauge Pointer Cap

C1RadialGauge has the special Cap property (C1GaugeCap). It draws a filled circle or several filled circles on top of the gauge pointers. It may appear behind the pointers if the BehindPointers property is set to **True**.

The main circle appears behind other circles that are added to the **C1GaugeCap**.**MoreCircles** collection property. Main circle may have a border while the other circles have only filling. You may also specify whether the custom image should appear instead of circles in the pointer cap.

## Gauge Decorators

Decorators graduate the range of possible values. They are derived from the **C1GaugeDecorator** class. The following is the list of available decorators:

- C1GaugeLabels – draws a sequence of labels on the gauge. Labels have the Format property that specifies the numeric format. To prepend or append values with arbitrary text use the custom numeric format. If this doesn't help, the FormatLabel event gives unlimited options for the value formatting. Use the Name property to distinguish label decorators in the event handler. The **C1GaugeLabels.FontSize** property should be set to some non-default value to scale the font size if the gauge's dimension changes.
- C1GaugeSingleLabel – this is a single label with almost same options as C1GaugeLabels. It has the **C1GaugeSingleLabel.Value** property that can be bound to a pointer using the PointerIndex property. Also, you can assign fixed text to a single label using the Text property. The Angle and Position properties give an ability to display the label at an arbitrary position.
- C1GaugeMarks – displays a sequence of tick marks on the gauge. There are a few predefined shapes for the marks that can be selected using the Shape property. Also, you can specify a custom shape using the

CustomShape property or display an image for each mark when using the CustomImage property.
- C1GaugeSingleMark – displays a single mark at the given position.
- C1GaugeRange – shows a range on the gauge. The range may have constant or variable locations and widths. It can be filled with simple or value-dependent gradient.

## Decorator Scales

There is actually one scale for the whole gauge. It has the fixed minimum and maximum values (see the Minimum and Maximum properties). Neither of decorators can display values out this range. Labels have the **ValueOffset** and **ValueFactor** properties that affect conversion of label values to the corresponding text and thus simulate changing the scale.

The Interval property sets the value interval to draw each tick mark or label. The IntervalWidth property specifies the distance between tick marks or labels in logical coordinates. You should assign one of these properties to graduate the scale. The difference between Interval and IntervalWidth appears if you resize the gauge. The first property maintains the constant value difference between the near tick marks while the actual distance in pixels or logical coordinates may vary. The second property maintains the constant distance regardless of the value difference. This is useful for elements such as a stacked progress bar where the individual tick marks behave as decorative items whose values are not important. The Interval and IntervalWidth properties can be negative to start graduation from the maximum value.

The From and To properties set the value interval where the decorator can appear. The C1GaugeRange decorator has also these properties. So you can, for example, display a range in the given subinterval, not for the whole scale. The FromPointerIndex and ToPointerIndex allow binding the **C1GaugeMultivalueDecorator.From** and **C1GaugeMultivalueDecorator.To** properties to a pointer. Thus, for example, the range's upper bound may depend on the current pointer's value. The same technique works for labels and tick marks as well. The **C1GaugeMultivalueDecorator.FromPointerIndex** and **C1GaugeMultivalueDecorator.ToPointerIndex** properties specify an index in the MorePointers collection. If you want to bind to the main pointer assign a large value to these properties, for example, 10000. Setting to a negative value cancels binding.

The ScaleFrom and ScaleTo properties specify where the decorator scale starts and ends. These properties are necessary because the **C1GaugeMultivalueDecorator.From** and **C1GaugeMultivalueDecorator.To** values may vary if they are bound to pointers while the graduation should often be fixed. Also, the location and width of a decorator may change linearly from **ScaleFrom** to **ScaleTo** values. By default, these properties have the same values as the **From** and **To** properties. It may occur that the **From** and **To** values don't coincide with any scale marks. It is possible to display such values using the ShowIrregularFrom and ShowIrregularTo properties.

A few label or tick mark decorators can be grouped into the sequence. The first decorator in the sequence shows all its values. The second decorator shows all values except those appeared on the first decorator. Each the next decorator shows all values that do not occur in any of the previous decorators. The SequenceNo property specifies the sequence number. C1GaugeLabels and C1GaugeMarks have independent sequences. You may exclude the decorator from any sequences by setting its SequenceNo properties to -1.

## Decorator ValueColors

All decorators have the **C1GaugeDecorator.ValueColors** collection property. It allows mapping colors to value thresholds to display, for example, a multi-colored range or a set of labels. A few gradient blending types are supported via the ValueColorFalloff property.

The **C1GaugeDecorator.ValueColors** collection contains objects of the C1GaugeValueColor type. Each of these objects associates a value (specified by the **C1GaugeValueColor.Value** property) with some color (specified by the Color and Opacity properties). Instead of the fixed value you may provide the index of a pointer that gives the value associated with this object. Thus, the **C1GaugeValueColor.PointerIndex** property allows binding the color to a pointer.

For filled decorators, such as ranges or tick marks, the ValueColors property is used if the BrushType property equals

to 'SolidColor'.

If several decorators use the same color mapping you can create the common mapping item (**CommonColorMap**) in the **C1GaugeBase.ColorMaps** collection. The common item includes the **CommonColorMap.ValueColors** and **CommonColorMap.ValueColorFalloff** properties. The **CommonItem.Name** property defines a name that can be assigned to the **C1GaugeDecorator.ColorMapName** property. Then, the common mapping colors will be used instead of **C1GaugeDecorator.ValueColors**.

## Marker ValueImages

**C1GaugeMarks** and **C1GaugeSingleMark** have the **CustomImage** property that associates a custom image with the tick marks. Also, it's possible to associate several images with the tick marks. The **ValueImages** property of **C1GaugeMarks** and **C1GaugeSingleMark** specifies the collection of values and their associated images. If the value of the tick mark is more or equal to **C1GaugeValueImage.Value** the image from **C1GaugeValueImage.CustomImage** appears on the tick mark. The **C1GaugeValueImage.PointerIndex** property allows binding the image to a gauge pointer.

If there are a few tick marks with the same set of images you can use the **C1GaugeBase.ImageMaps** property to create a common template (**CommonImageMap**) for the **ValueImages** collection. After that, you can assign the template name to the **ImageMapName** property of **C1GaugeMarks** or **C1GaugeSingleMark**. Then, the common template will be used instead of the decorator's **ValueImages** property.

## Decorator Layout

There are a number of layout options available for decorators. The Location property sets the distance, in logical coordinates, between the decorator and the center point (for radial gauges) or the transversal axis start (for linear gauges). The Location2 property specifies the same distance at the maximum value to change it linearly along the value axis.

The Alignment and **OrthogonalAlignment** properties specify how the decorator is aligned in both directions. Labels and tick marks can be rotated depending on their value on a radial gauge if their **IsRotated** property equals to **True**. If labels appear inverted for some values they can be flipped if the value of the **AllowFlip** property is **True**. The **TextAngle** (for labels) and **ShapeAngle** (for tick marks) properties specify an additional angle that is used to rotate items on both linear and radial gauges.

The labels are not scaled by default, i.e. the em-size of the font is defined by the **Font** property of **C1GaugeSingleLabel** or **C1GaugeLabels**. If you want to scale labels like other gauge elements assign a numeric value to the **FontSize** property. Also, in a **C1GaugeLabels**, you can vary the font size linearly along the value axis using the **C1GaugeLabels.FontSize** to **C1GaugeLabels.FontSize2** properties.

If several labels use the same font color, size, and other settings you can create a common font object in the **CommonFonts** collection of **C1Gauge** or **C1GaugeBase**. Then, you may reference the common font from a **C1GaugeSingleLabel** or **C1GaugeLabels** using the **CommonFontName** property.

## Gauge Borders and Filling

All gauge elements, except text labels and images, have the following properties in the Appearance category: **Border** (C1GaugeBorder), **Filling** (C1GaugeFilling), **Gradient** (C1GaugeGradient).

Border specifies the color, thickness, and style of the pen that draws the element border. The default border color is the same as **ForeColor** of the owner C1Gauge control. To hide the border set its **C1GaugeBorder.LineStyle** property to None. To display a semitransparent border specify the value of alpha when setting the **C1GaugeBorder.Color** property, for example: "128, 0, 0, 0".

The **Filling** property specifies how to draw the interior. BrushType is the main property of the C1GaugeFilling class.

The filling color is defined by a pair of properties: Color and Opacity. Some brushes use the second pair of color properties as well: Color2, Opacity2. You may switch these pairs using the SwapColors property. The following are possible values for the BrushType property:

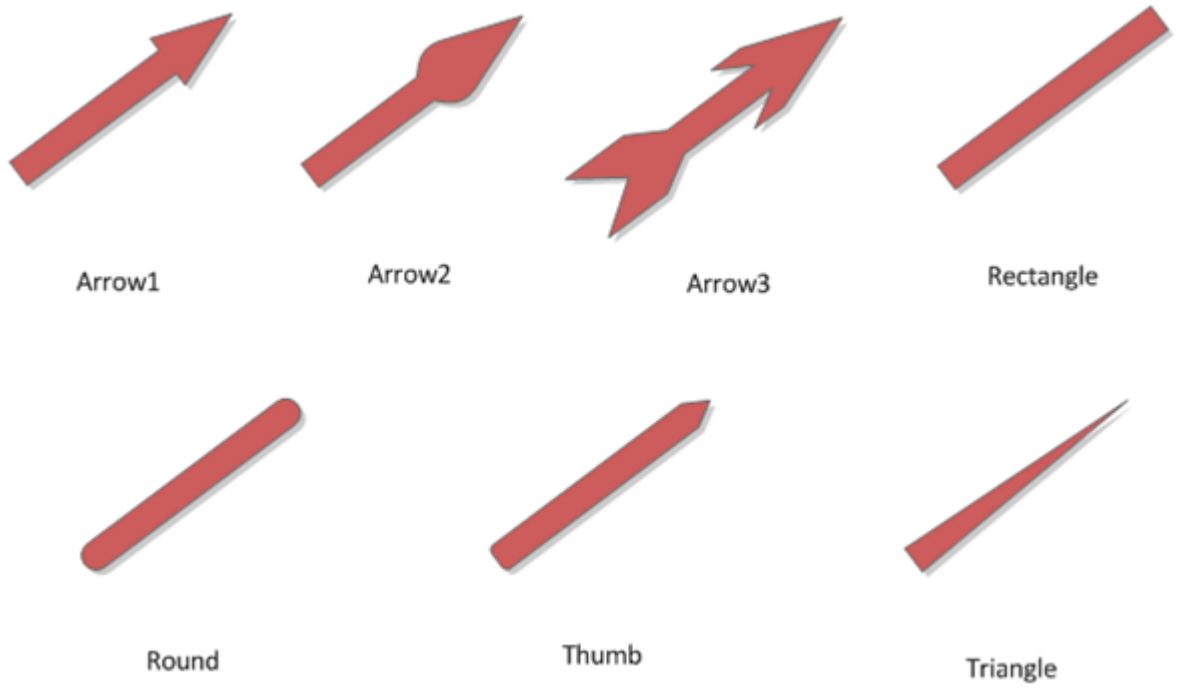
- **None** – the element has no filling.
- **SolidColor** – the element is filled with a single color specified by the **Color** property. If this property is empty, the **BackColor** property of the owner C1Gauge control is used instead.
- **Hatch** – fills the element with one of the predefined hatch styles (selected by the HatchStyle property) using a foreground color (Color) and a background color (Color2).
- **Texture** – fills the interior of an element using an image provided in the **C1GaugeFilling.TextureImage** property. The wrap mode for the texture brush is specified by the **C1GaugeFilling.WrapMode** property.
- **Gradient** – the interior is filled with a gradient that is defined by the **Gradient** property (see below).

The **Gradient** property provides a few settings for drawing a gradient. The **C1GaugeGradient.Direction** property sets the direction of the gradient orientation line, such as Vertical, Radial, and so on. The **RadialInner** value draws the gradient that is inscribed into the owner element. The **RadialOuter** value draws the circumscribed gradient. The **C1GaugeGradient.Falloff** property allows using the bell-shaped or triangular gradient effects. The **C1GaugeGradient.TranslateX** and **C1GaugeGradient.TranslateY** properties allow moving the area filled with the gradient by the specified relative amount, and then the **C1GaugeGradient.ScaleX** and **C1GaugeGradient.ScaleY** properties scale this area, if necessary.

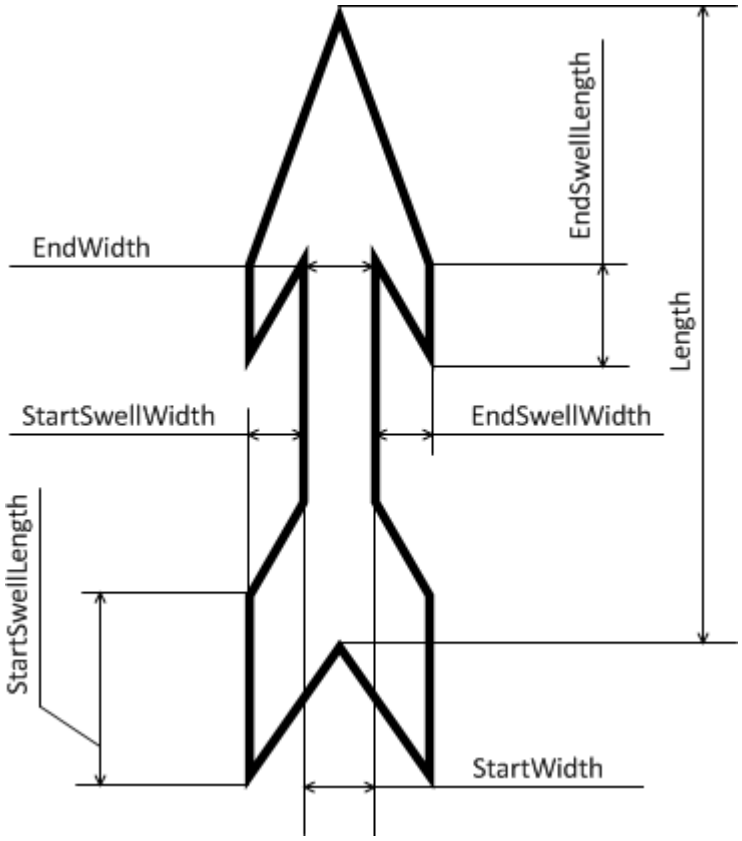
If several elements have similar borders, fillings, or gradients you can create the common templates that can be shared between several objects. Both **C1Gauge** and **C1GaugeBase** classes have the special collections of common items: **CommonBorders**, **CommonFillings**, and **CommonGradients**. Each common item, such as a **CommonBorder** and **CommonFilling**, has the **Name** property. If this property is not empty you can choose the specified name from the drop-down list that opens for the **Border**, **Filling**, and **Gradient** properties of various gauge elements. To reference a common item from code, you should assign the item name to the following properties: **C1GaugeBorder.CommonBorderName**, **C1GaugeFilling.CommonFillingName**, **C1GaugeGradient.CommonGradientName**.

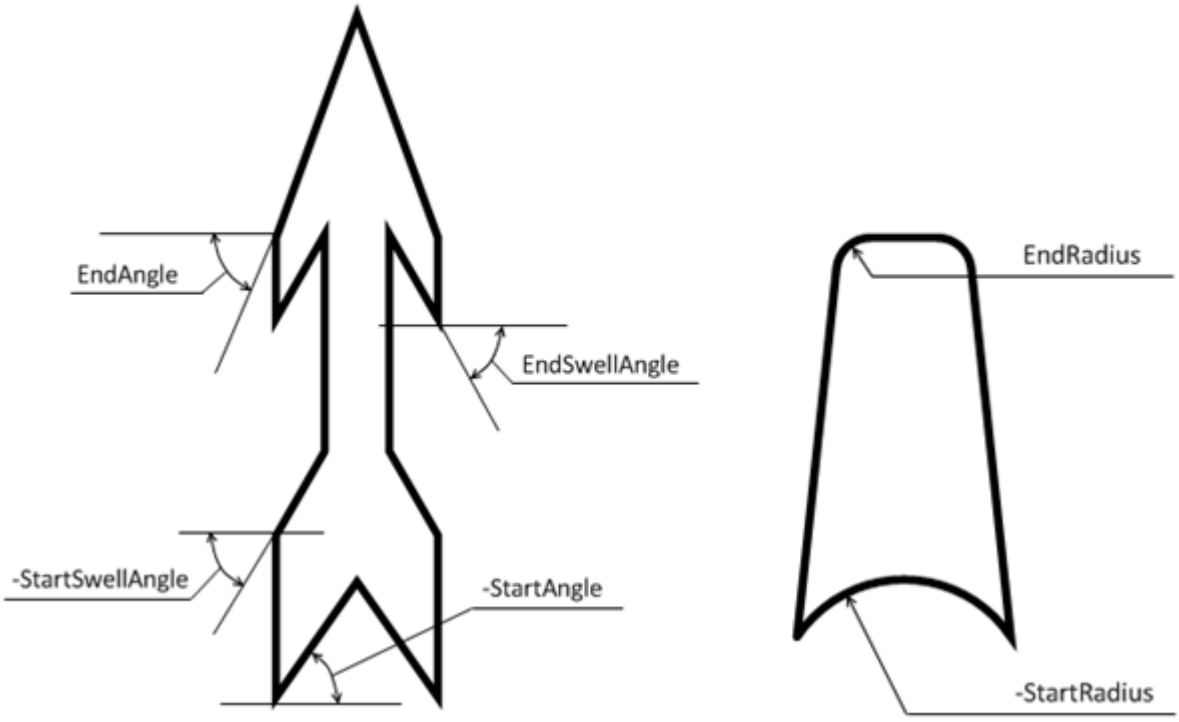
## Custom Pointers and Mark Shapes

Pointers and tick marks use the special class (C1GaugeCustomShape) that draws as a geometric figure with a number of layout options. Also, there are a few predefined shapes. The following are the predefined shapes available via the **C1GaugePointer**.**Shape** property:




The properties of a custom shape are specified in **C1GaugePointer**.**CustomShape**. The next images illustrate all the available options. All angles and radiuses can be either positive or negative. Here the **Length** property belongs to C1GaugePointer, not to C1GaugeCustomShape. There is also the **Width** property in C1GaugePointer that sets the maximum of **C1GaugeCustomShape.StartWidth** and **C1GaugeCustomShape.EndWidth**. The **C1GaugePointer.FlipShape** property allows flipping the pointer. So the start becomes end and vice versa.






The **C1GaugeSingleMark** and **C1GaugeMarks** decorators have the **Shape** and **CustomShape** properties as well. 'Rectangle', 'Round', and 'Triangle' are the only available predefined shapes for the tick marks.

It may occur that several gauge elements, for example a **C1GaugeMarks** and a **C1GaugePointer**, use the same custom shape (**C1GaugeCustomShape**). If so, you can create a common shape in the **CommonShapes** collection of the owner **C1Gauge** or **C1GaugeBase**. Then you can select the name of the common shape from the drop-down list of the **CustomShape** property or you can assign this name to the **C1GaugeCustomShape.CommonShapeName** property from code.

## Custom Pointer, Mark, and Cap Images

If neither of predefined shapes nor the custom shapes are suitable for your needs use the **CustomImage** property that allows to specify an image for using as a pointer, pointer cap, or tick mark.

The Image is the main property of the C1GaugeCustomImage object. It can be assigned to a bitmap or metafile that will appear as a pointer or decorator. The **Width** and **Height** properties allow resizing the source image. The **RotateFlipType** property gives an ability to rotate and flip the source image. The **Hue**, **Lightness**, **Saturation**, and **Opacity** property modify the HLS and alpha settings of the source image. Actually, neither of these operations affects the source image assigned to the Image property. They work with a copy of the source image.

The **KeepAspectRatio** and **KeepSize** properties impose a restriction on how the custom image can be resized in the target element, such as the pointer or the marks decorator. If the **KeepSize** property is **True** the **Length** and **Width** of the pointer, for example, don't affect the image size. If the **KeepSize** is **False** while the **KeepAspectRatio** property is **True** the image height may vary depending on the **Length** property of the target element. The image width changes correspondingly to maintain the aspect ratio of the custom image.

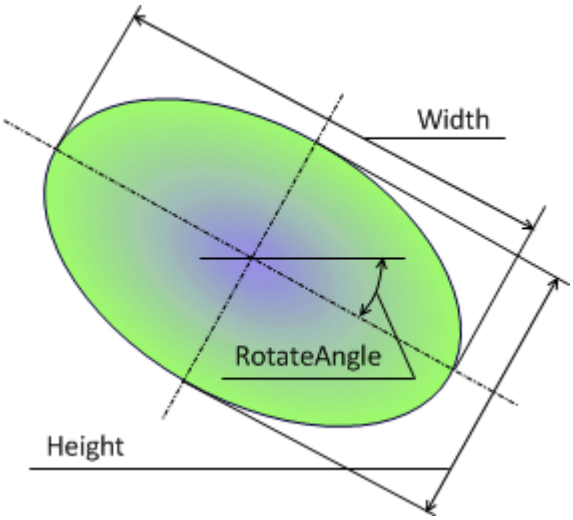
It may be a good idea to collect all images in the **C1Gauge.CommonImages** and **C1GaugeBase.CommonImages** properties. You can reference an item from the **CommonImages** collection by assigning its name to the **C1GaugeCustomImage.CommonImageName** property or by selecting a common image name from the drop-down list of the **CustomImage** property editor in the property grid.

## Gauge Face and Cover Shapes

Like a watch or thermometer, the gauge controls include a **Face** and a **Cover**. The **Face** appears above the background but behind the pointer and decorators, and the **Cover** appears, like a glass over a thermometer, above all other elements. Shapes can be added to the **FaceShapes** and **CoverShapes** collections of the C1Gauge and C1GaugeBase objects. Each the shape has the **CenterPointX** and **CenterPointY** properties which define its position in the working area. The default value of these properties is 0.5 (the center of the shape appears in the center of its viewport). Shapes can't have shadow.

## Ellipse

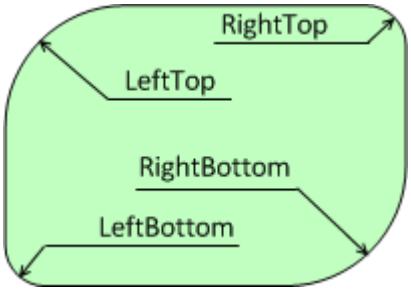
C1GaugeEllipse is the simplest shape. When filling the ellipse with radial gradient set the gradient's **Direction** property to **RadialInner**. If the **Width** (**Height**) property is positive it uses the same logical coordinates as gauge pointers and decorators. If the **Width** (**Height**) property is negative its value specifies the portion of the owner element's width (or height). So, -1 is the whole width (height) of the owner element, -2 is double width (height) of the owner element, and so on.



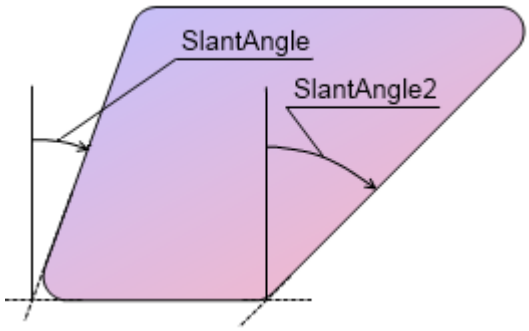

[Settings for this shape: Width = -1, Height = -0.6, RotateAngle = 20]

## Rectangle

The C1GaugeRectangle shape draws a rectangle with optionally rounded corners. The default corner radius is specified by the **CornerRadius** property. Other settings are the same as for C1GaugeEllipse. When filling a rectangle with radial gradient set the **C1GaugeGradient**.**Direction** property to **RadialOuter**.



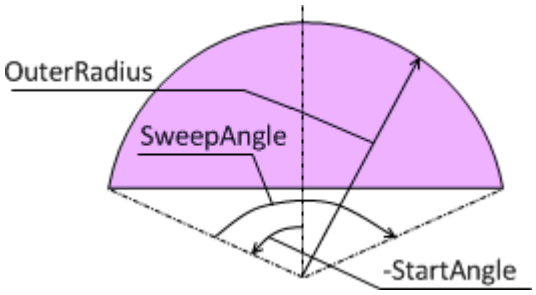

[Settings for this shape: Width = -1, Height = -0.7, CornerRadius = 20, LeftTop = 40, RightBottom = 40]



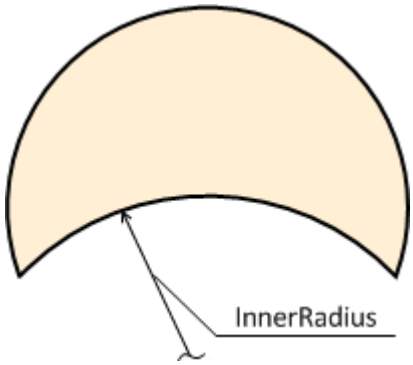

[Settings for this shape: SlantAngle = 20, SlantAngle2 = 45]

## Segment

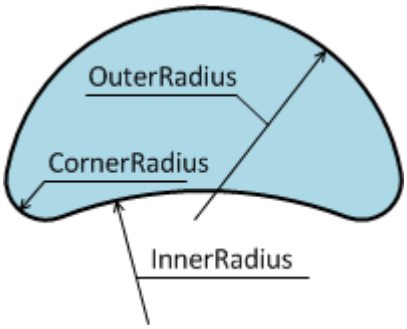
C1GaugeSegment shows a part of a circle as it is shown at the below images. The **StartAngle** and **SweepAngle** properties are in degrees. They don't take the **CornerRadius** into account (work as if the **CornerRadius** is 0). The **InnerRadius** property can be positive, negative, or 0 (default).



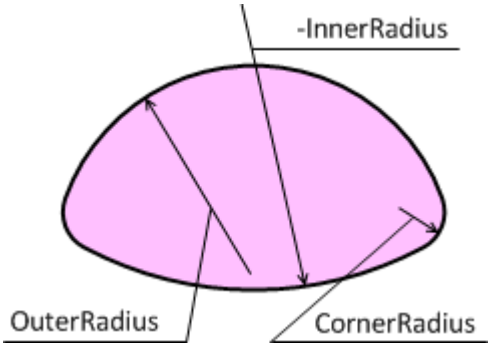

[Settings for the above shape: StartAngle = -80, SweepAngle = 160, OuterRadius = 100]




[Settings for the above shape: StartAngle = -115, SweepAngle = 230, OuterRadius = 100, InnerRadius = 110]



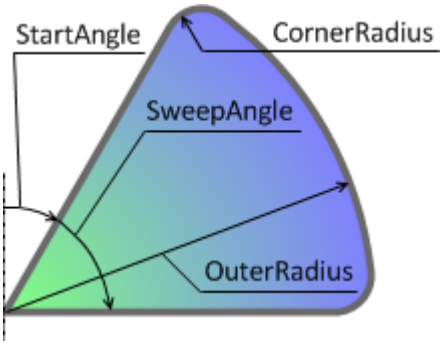

[Settings for the above shape: StartAngle = -115, SweepAngle = 230, CornerRadius = 20, OuterRadius = 100, InnerRadius = 160]



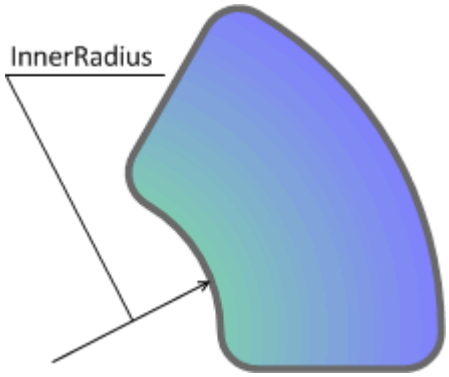

[Settings for the above shape: StartAngle = -80, SweepAngle = 160, CornerRadius = 20, OuterRadius = 100, InnerRadius = -200]

## Sector

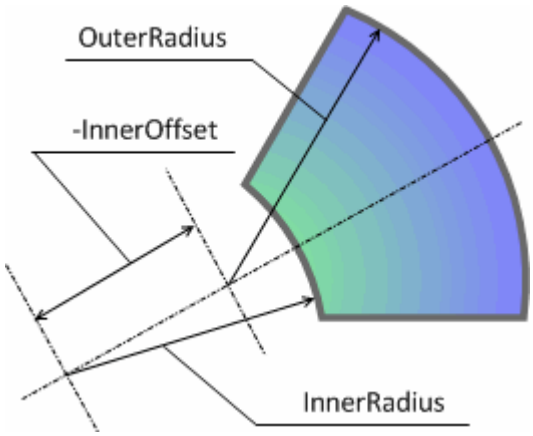
C1GaugeSector is the most complex shape. You can observe all the available options in the following images.



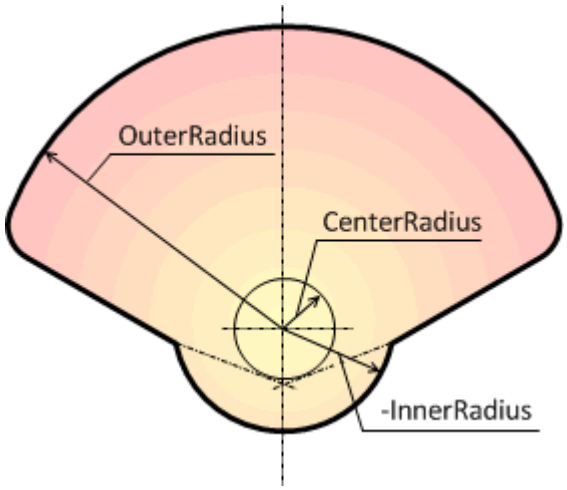

[Settings for the above shape: StartAngle = 30, SweepAngle = 60, CenterRadius = 0, CornerRadius = 10, InnerRadius = 0, OuterRadius = 100]




[Settings for the above shape: StartAngle = 30, SweepAngle = 60, CenterRadius = 0, CornerRadius = 10, InnerRadius = 45, OuterRadius = 100]




[Settings for the above shape: StartAngle = 30, SweepAngle = 60, CenterRadius = 0, CornerRadius = 0, InnerOffset = -60, InnerRadius = 100, OuterRadius = 100]




[Settings for the above shape: StartAngle = -70, SweepAngle = 140, CenterRadius = 20, CornerRadius = 10, InnerRadius = -35, OuterRadius = 100]

## Caption

C1GaugeCaption is an element that draws fixed text. The **Width** and **Height** properties can restrict the bounds of the

caption. Set the **FontSize** property to some value other than default to scale the font size when the owner element size changes.


[Settings for this caption: Width = -0.5, Height = -0.4, FontSize = 10, Alignment = Far, RotateAngle = -20]

## Image

C1GaugeImage draws an image on the gauge. The image size can be changed using the **Width** and **Height** properties (if the **KeepSize** property is False). The aspect ratio will be maintained if the **KeepAspectRatio** property is equal to True. Image can be rotated with the **RotateAngle** property. You may change opacity of the image and its HSL settings using the **Opacity**, **Hue**, **Saturation**, and **Lightness** properties. The **FlipType** property set the axis used to flip the image. For example:

[Settings for this image: RotateAngle = 20]

## Clippings

The filled shapes can also be used for clipping. The **Clippings** collection property is available in the following elements: C1GaugeBaseShape, C1GaugePointer, C1GaugeDecorator, and C1GaugeCap. To specify the clipping region for some element you may follow this:

- Assign a string to the **C1GaugeBaseShape.Name** property of the shape or shapes that will be used for clipping.
- Open the collection editor for the **Clippings** property of an element that is being clipped.
- Add one or several items to the **Clippings** collection and set their **C1GaugeClipping.ShapeName** properties in accordance with names assigned at the first step. Also, assign the **C1GaugeClipping.Operation** property to some non-default value (**Intersect**, for example).

The **C1GaugeClipping.ScaleFactor** property allows scaling of the clipping region. You may leave the **C1GaugeClipping.ShapeName** property empty for a filled shape. Then it will be clipped by itself. So if you set the **C1GaugeClipping.ScaleFactor** to 0.9, for example, you will see a thick border instead of the filled shape.

The name of the clipping shape is searched in the **C1GaugeBase.FaceShapes** and **C1GaugeBase.CoverShapes** collection of this gauge. If it is not found in the current gauge it is then searched in the owner C1Gauge control.

## User Interaction

All visible gauge elements, such as pointers, decorators, caps, shapes, have now the **HitTestable** property. This property affects both design-time and runtime behavior of **C1Gauge**.

## Design-time Interaction

**C1Gauge** paints the topmost hit-testable element under the mouse pointer using the special brush. You can specify the color, opacity, and the hatch style of this brush using the **C1Gauge.HotBrush** property. The special tooltip appears under the **C1Gauge** control. It shows the type, name, and "path" to the "hot" element.

If you click the highlighted element its owner gauge (**C1RadialGauge**, **C1LinearGauge**, or the container control – **C1Gauge**) becomes selected on the designer surface. You can double click the "hot" gauge element to display a popup window with properties of the given element. As alternative way, you may right-click the "hot" element, then select "Launch Item Editor" from its context menu. Again, this only works for gauge items which have the **HitTestable** property set to True. Other elements behave as "transparent".

Clicking at the point where there is no any hit-testable element selects the gauge component (**C1RadialGauge** or **C1LinearGauge**) or the container control (**C1Gauge**). A double click in **C1Gauge** or **C1GaugeBase** opens its editor in a popup window.

## Run-time Interaction

There are a number of events that occur for **C1Gauge** and **C1GaugeBase** components as the result of various user actions at runtime. The source of these events is a hit-testable element that can be obtained from the **Item** property of the **ItemEventArgs** object passed to the event handlers.

- **ItemStateChanged** – fires when a hit-testable item becomes enabled, hot, pressed, or vice-versa.
- **ItemClick** – fires when a gauge element is clicked.
- **ItemDoubleClick** – fires when a gauge element is double-clicked.
- **ItemMouseEnter** – fires when the mouse pointer enters a gauge element.
- **ItemMouseLeave** – fires when the pointer leaves a gauge element.
- **ItemMouseMove** – fires when the mouse pointer is moved over a gauge element.
- **ItemMouseDown** – fires when the pointer is over a gauge element and a mouse button is pressed.
- **ItemMouseUp** – fires when the mouse pointer is over a gauge element and a mouse button is released.

The **ItemStateChanged** event can be used for updating appearance of a gauge item when it becomes hot or pressed. After setting **C1Gauge.SupportsTransitionEffect** to True you can apply the special transition effect when changing the state of a gauge item. For example:

**To write code in Visual Basic**

Visual Basic

```
Private Sub c1LinearGauge1_ItemStateChanged(ByVal sender As System.Object, _
  ByVal e As ItemEventArgs) Handles c1LinearGauge1.ItemStateChanged
      If TypeOf e.Item Is C1GaugePointer Then
            Dim p As C1GaugePointer = CType(e.Item, C1GaugePointer)
            c1Gauge1.BeginUpdate()

            If e.ItemPressed Then
```

```
                    p.Filling.CommonFillingName = "pressedFilling"
            ElseIf e.ItemHot Then
                    p.Filling.CommonFillingName = "hotFilling"
            Else
                    p.Filling.CommonFillingName = "normalFilling"
            End If

            c1Gauge1.EndUpdate(200)
      End If
End Sub
```

**To write code in C#**

C#

```
private void c1LinearGauge1_ItemStateChanged(object sender, ItemEventArgs e)
{
      if (e.Item is C1GaugePointer)
      {
            C1GaugePointer p = e.Item as C1GaugePointer;
            c1Gauge1.BeginUpdate();

            if (e.ItemPressed)
                  p.Filling.CommonFillingName = "pressedFilling";
            else if (e.ItemHot)
                  p.Filling.CommonFillingName = "hotFilling";
            else
                  p.Filling.CommonFillingName = "normalFilling";

            c1Gauge1.EndUpdate(200);
      }
}
```

The **ItemMouseDown** and **ItemMouseMove** events can be used for updating the pointer value. For example:

**To write code in Visual Basic**

Visual Basic

```
Private Sub c1RadialGauge1_ItemMouseDown(ByVal sender As System.Object, _
  ByVal e As ItemMouseEventArgs) Handles C1RadialGauge1.ItemMouseMove,
C1RadialGauge1.ItemMouseDown
If (e.Button And MouseButtons.Left) = MouseButtons.Left Then
 Dim p As C1GaugePointer = e.Gauge.Pointer
 p.Value = p.GetValueAt(e.X, e.Y)
End If
End Sub
```

**To write code in C#**

C#

```
private void c1RadialGauge1_ItemMouseDown(object sender, ItemMouseEventArgs e)
{
```

```
 if ((e.Button & MouseButtons.Left) == MouseButtons.Left)
  {
   C1GaugePointer p = e.Gauge.Pointer;
   p.Value = p.GetValueAt(e.X, e.Y);
  }
}
```

In the above sample we convert the mouse position to a value using the **C1GaugePoiner.GetValueAt()** function. It may be difficult for the user to specify any concrete value using this method. To facilitate it we can round the value to the nearest multiple of the given step (snapInterval) using the **C1GaugePointer.UpdateValue()** method instead setting a value to the **C1GaugePointer.Value** property directly. Then, the next line

```
p.Value = p.GetValueAt(e.X, e.Y)
```

should be replaced with something like the following:

```
p.UpdateValue(p.GetValueAt(e.X, e.Y), 1.0)
```

After that, the assigned value will be rounded automatically to integer numbers.

There are a few additional events in the C1GaugeBase component. They allow interaction with gauge pointers.

- **C1GaugeBase.PointerDragBegin** – fires when a user starts dragging a gauge pointer.
- **C1GaugeBase.PointerDragEnd** – fires when the user ends dragging a gauge pointer.
- **C1GaugeBase.PointerDragMove** – fires when a gauge element is dragged with the mouse.
- **C1GaugeBase.PointerDragCancel** – allows resetting the pointer value to its original state.

The **C1GaugeBase.PointerDragMove** event allows simple updating of the pointer value. For example:

**To write code in Visual Basic**

Visual Basic

```
Private Sub c1RadialGauge1_PointerDragMove(ByVal sender As System.Object, _
  ByVal e As PointerDragEventArgs) Handles c1RadialGauge1.PointerDragMove
e.Pointer.UpdateValue(e.NewValue, 0.25)
End Sub
```

**To write code in C#**

C#

```
private void c1RadialGauge1_PointerDragMove(object sender, PointerDragEventArgs e)
{
e.Pointer.UpdateValue(e.NewValue, 0.25);
}
```

The same handler can be attached to the **C1GaugeBase.PointerDragCancel** event. To use this event please make sure that the **C1Gauge.Selectable** property equals to True. If the user is dragging the pointer and decides to cancel this change and return to the previous value she can press the ESC key. Then, the **PointerDragCancel** event will be fired with the **PointerDragEventArgs.NewValue** property set to the previous value.

It is now possible to use the keyboard events, such as **Control.KeyDown**, **Control.KeyPress**, and others, with the **C1Gauge** control after setting the **C1Gauge.Selectable** property to True. You can also attach a handler to the

**C1Gauge.DrawFocus** event. This gives ability to change the bounds of the focus rectangle or draw your own focus selection.

## Shadows

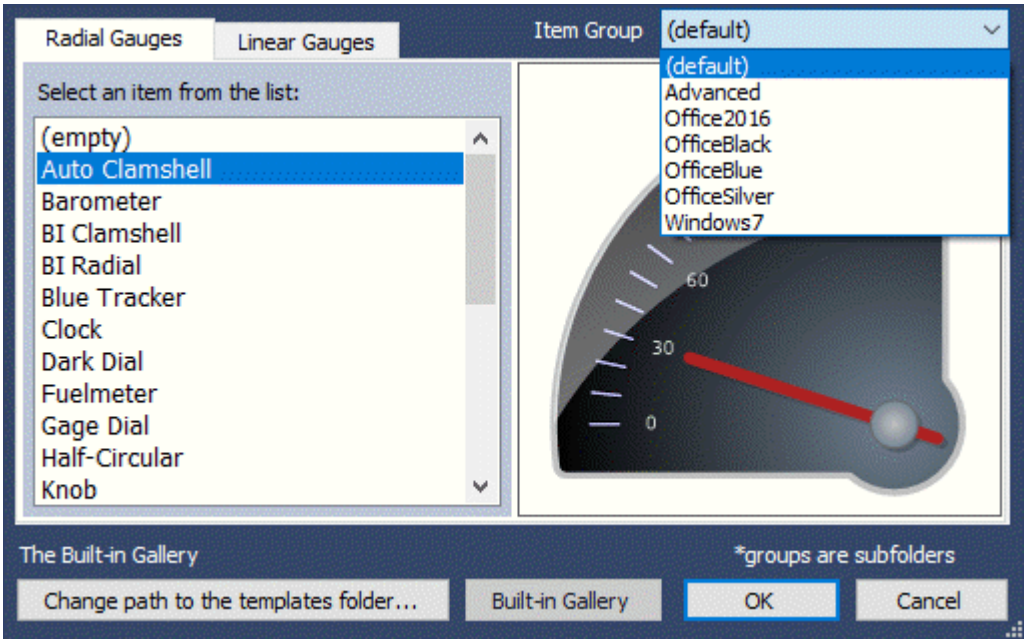
The following elements can have shadow: **C1GaugePointer**, **C1GaugeDecorator**, and **C1GaugeCap**. They have the **Shadow** property of the **C1GaugeShadow** type whose settings are inherited by default from the owner **C1Gauge** control's **Shadow** object except the **C1GaugeShadow.Visible** property. It is False for all individual elements, by default. You may specify the **C1GaugeShadow.Color** and **C1GaugeShadow.Opacity** of shadow. The **C1GaugeShadow.OffsetX** and **C1GaugeShadow.OffsetY** properties set the logical offset of shadow relatively to its owner element.

Instead of setting various properties of each **C1GaugeShadow** object you can create a common shape in the **CommonShapes** collection of **C1Gauge** or **C1GaugeBase**. This common shape can be referenced from multiple gauge elements. Just select its name from the drop-down list of the **Shadow** property or assign its name to the **C1GaugeShadow.CommonShadowName** property from code.

The offset properties for all element shadows are set in logical coordinates in the container control scope. The offset doesn't depend on individual gauge's logical coordinates; all shadows are of the same size by default.

# Gauges for WinForms Appearance

Select the C1Gauge control in your project and click the smart tag. Select **Add New Gauge** from the **C1Gauge Tasks** menu. The **New Gauge Gallery** dialog box appears, allowing you to pick and choose templates and template groups:



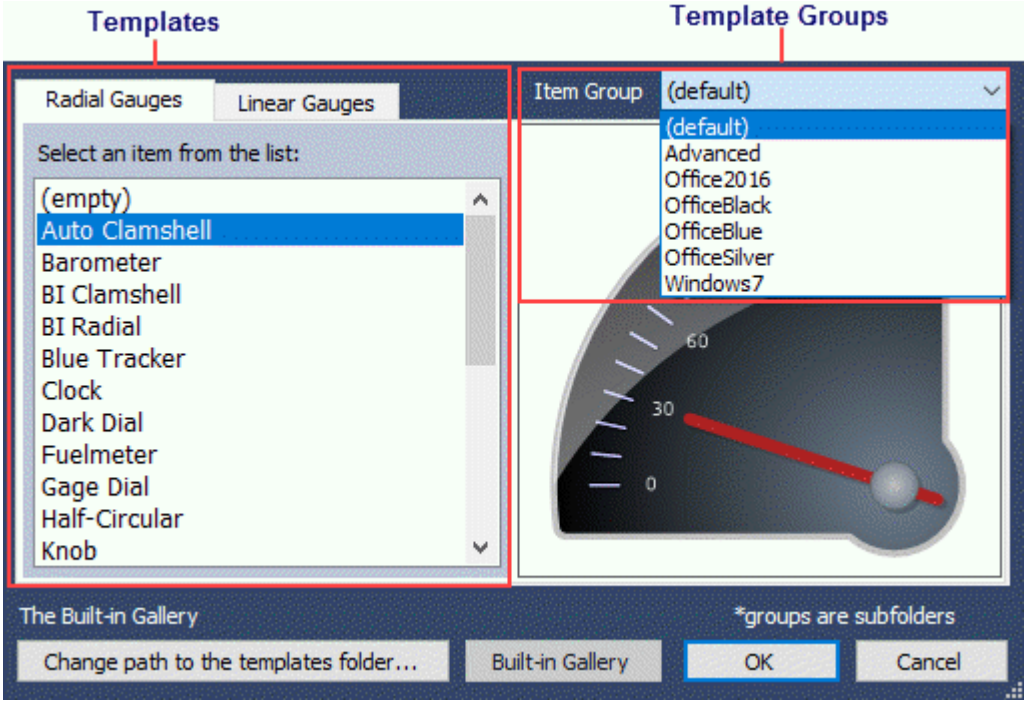

The following topics explain how to configure elements of **Gauges for WinForms**' display, such as built-in template groups, templates, and appearances.

# C1RadialGauge Templates and Template Groups

**Gauges for WinForms** provides a wide variety of built-in gauge templates and six template groups, or Item Groups, which provide the available templates for the group you choose.

To access the C1RadialGauge templates and template groups, select the C1Gauge control on your form and click the smart tag.

Select **Add New Gauge** from the **C1Gauge Tasks** menu. The **New Gauge Gallery** dialog box appears.



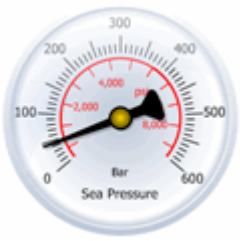

Click the **Radial Gauges** tab. You can select one of the built-in templates or choose **(empty)** to create your own.

To select a template group, click the drop-down list next to **Item Group**. Notice that the available templates for the group appear in the list under **Radial Gauges**.

## Template Groups

The following tables display the templates available for each of the six Item Groups.

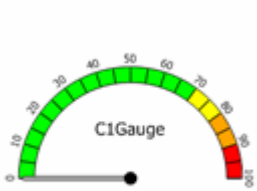
## Item Group: (default)

| | | |
|---|---|---|
|  **Auto Clamshell** |  **Barometer** |  **BI Clamshell** |
|  **BI Radial** |  **Blue Tracker** |  **Clock** |

Dark Dial

Fuelmeter

Gage Dial




Half-Circular

Knob

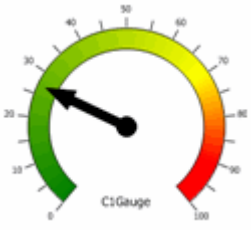
Multi Range


Needle Gauge

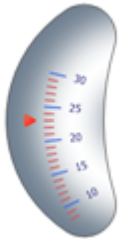
Radial Indicator

Radial Numeric

Range Dial




Simple Range



Sliding Scale

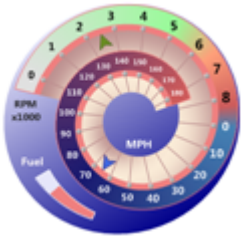

Soft Meter


Speedometer

Tachometer


Technical

## Item Group: Advanced

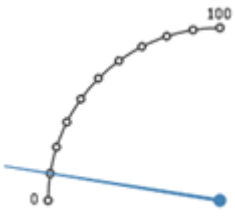
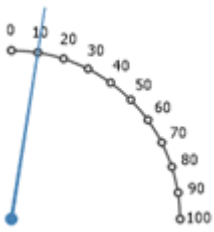
| | | |
|---|---|---|
|  |  |  |
| **Concept Car** | **Helical** | **Logarithmic** |

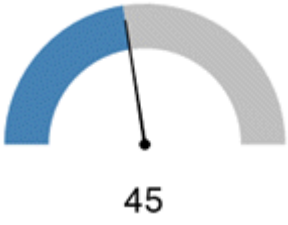
## Item Group: OfficeBlack, OfficeBlue, OfficeSilver

Note that the color of the template will be different, depending on which Office template group you choose.

| | |
|---|---|
|  |  |
| **Clamshell 1** | **Clamshell 2** |
|  |  |
| **Half Circle** | **Radial** |

## Item Group: Office2016

| | |
|---|---|
|  |  |
| **Clamshell 1** | **Clamshell 2** |

| | |
|---|---|
|  **Half Circle** |  **Radial** |

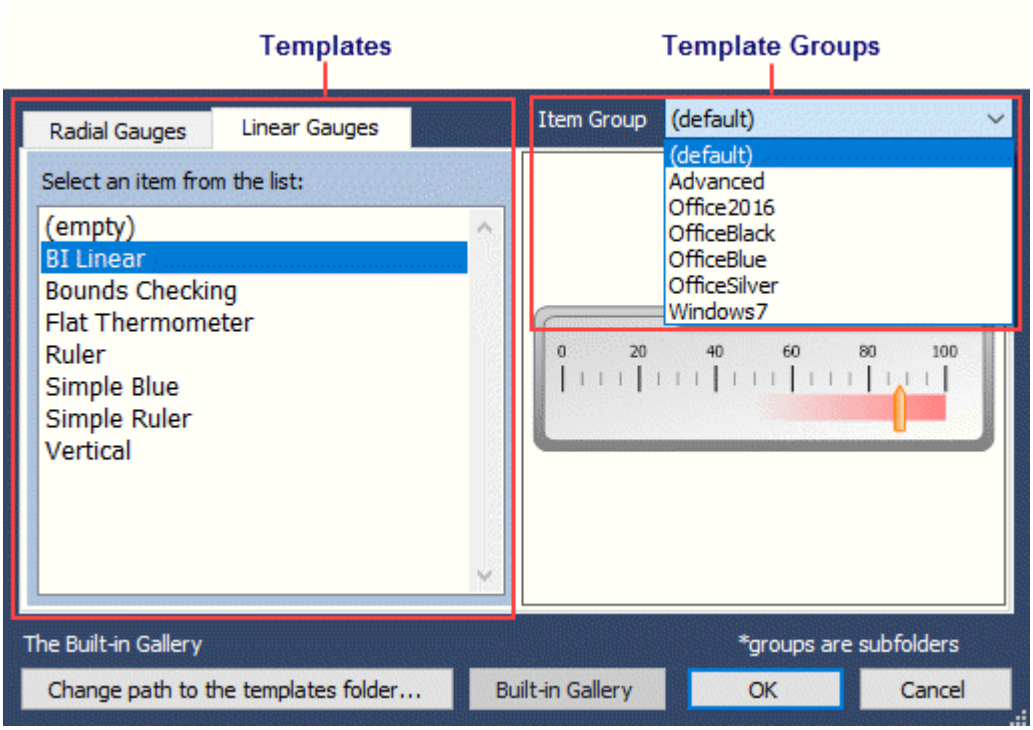
## Item Group: Windows 7

| | | |
|---|---|---|
|  **Clamshell 1** |  **Clamshell 2** |  **Clamshell 3** |
|  **Clamshell 4** |  **Half Circle 1** |  **Half Circle 2** |
|  **Radial 1** |  **Radial 2** | |

# C1LinearGauge Templates and Template Groups

To access the C1LinearGauge templates, select the C1Gauge control and click the smart tag. Select **Add New Gauge** from the **C1Gauge Tasks** menu.

The **New Gauge Gallery** dialog box appears, allowing you to pick and choose templates and template groups.



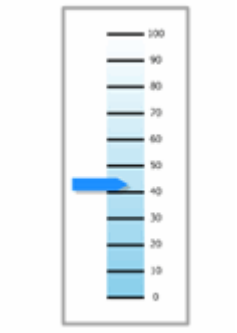
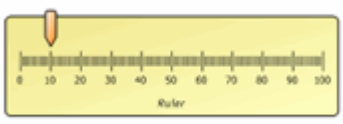

Click the **Linear Gauges** tab. You can select one of the built-in templates or choose **(empty)** to create your own.

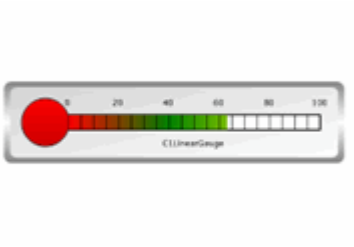
To select a template group, click the drop-down list next to **Item Group**. Notice that the available templates for the group appear in the list under **Linear Gauges**.

## Template Groups

The following tables display the templates available for each of the six Item Groups.

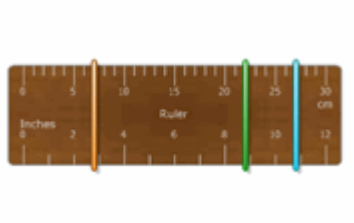
## Item Group: (default)

| | |
|---|---|
|  **BI Linear** |  **Simple Blue** |
|  **Bounds Checking** |  **Simple Ruler** |



Flat Thermometer

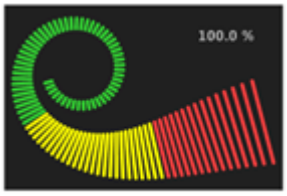
Vertical




Ruler

## Item Group: Advanced

Frequencies

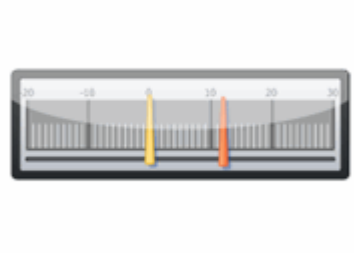



Lituus

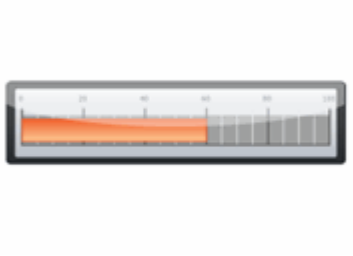

Thermometer

## Item Group: OfficeBlack, OfficeBlue, OfficeSilver

Note that the color of the template will be different, depending on which Office template group you choose.



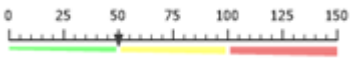
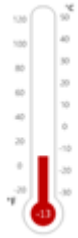
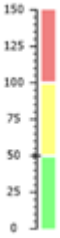
Horizontal



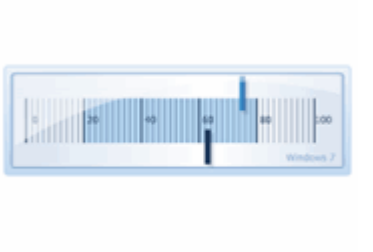
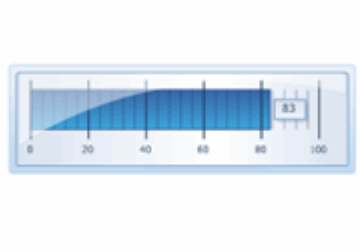
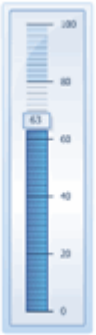
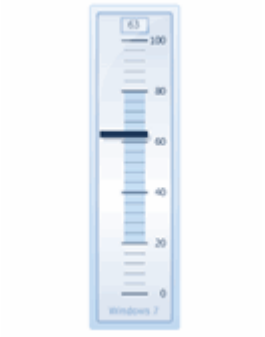
Progress Bar

| | |
|---|---|
|  Thermometer |  Vertical |

## Item Group: Office2016

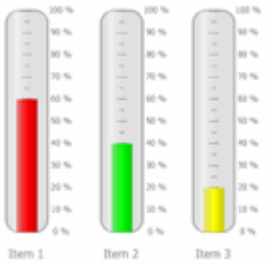
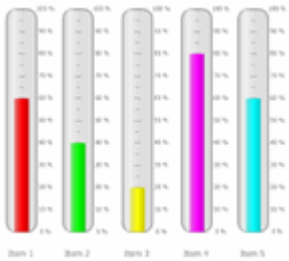
| | |
|---|---|
|  Horizontal |  Progress Bar |
|  Thermometer |  Vertical |

## Item Group: Window7

| | | |
|---|---|---|
|  Horizontal |  Progress Bar |  Thermometer |
|  Vertical 1 |  Vertical 2 | |

## Loading a C1Gauge from a Template

**Gauges for WinForms** allows you to load six different C1Gauge controls from templates. The templates include:

| | | |
|---|---|---|
|  **Composite** |  **Performance Monitor** |  **Radial Dashboard** |
|  **Test Tubes 3** |  **Test Tubes 5** |  **Dusk Blue** (available only when the Advanced Item Group is selected) |

To load an existing C1Gauge control from a template:

1. Select the C1Gauge control on your form and click the smart tag to open the **C1Gauge Tasks** menu.
2. Click **Load From Template**. The **Load C1Gauge From Template** dialog box opens.
3. Select one of the six templates and click **OK**. If you have a folder containing your own customized templates, you can access it by clicking the **Change path to the templates folder** button and selecting the folder.

## Saving a C1Gauge to an XML File

You can easily save a formatted C1Gauge control to an XML file using the following steps. If you only need to save the formatting and not the actual control, see Saving a C1Gauge View to an XML File.

1. Select the C1Gauge control on your form and click the smart tag to open the **C1Gauge Tasks** menu.
2. Click **Save to XML File**. The **Save C1Gauge To XML File** dialog box appears.
3. Enter a file name for the .xml and click **Save**. You can later load the gauge from the template. See Loading a C1Gauge from a Template for more information.

## Loading a C1Gauge View from an XML File

You can load predefined formatting for a C1Gauge control from an XML file. To do this, follow these steps:

1. Select the C1Gauge control on your form and click the smart tag to open the **C1Gauge Tasks** menu.
2. Click **Load Appearance**. The **Load c1Gauge1 View From XML File** dialog box appears.
3. Select the .xml file containing the formatting and click **Open**. The formatting is applied to your C1Gauge control.

## Saving a C1Gauge View to an XML File

You can edit the appearance of your gauge and then save it to an XML file, which can be loaded to format a C1Gauge control at a later time. Once you set up the gauge appearance as desired, follow these steps:

1. Select the C1Gauge control on your form and click the smart tag to open the **C1Gauge Tasks** menu.
2. Click **Save Appearance**. The **Save c1Gauge1 View To XML File** dialog box opens.
3. Enter a file name and click **Save**. If you do not enter a file name, the default file used will look similar to this: c1RadialGauge1.View.xml.

# Gauges for WinForms Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos, which may make use of other development tools included with the ComponentOne Studio.

Please refer to the pre-installed product samples through the following path:

**Documents\ComponentOne Samples\WinForms**

The following tables provide a short description for each sample.

| Sample | Description |
| --- | --- |
| GaugeDemo | This sample includes several examples of gauges, including both linear and radial gauges. |

## Gauges for WinForms Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio, and know how to use controls in general. If you are a novice to the **Gauges for WinForms** product, please see the Gauges for WinForms Quick Start first.

Each topic provides a solution for specific tasks using the **Gauges for WinForms** product. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of **Gauges for WinForms** features.

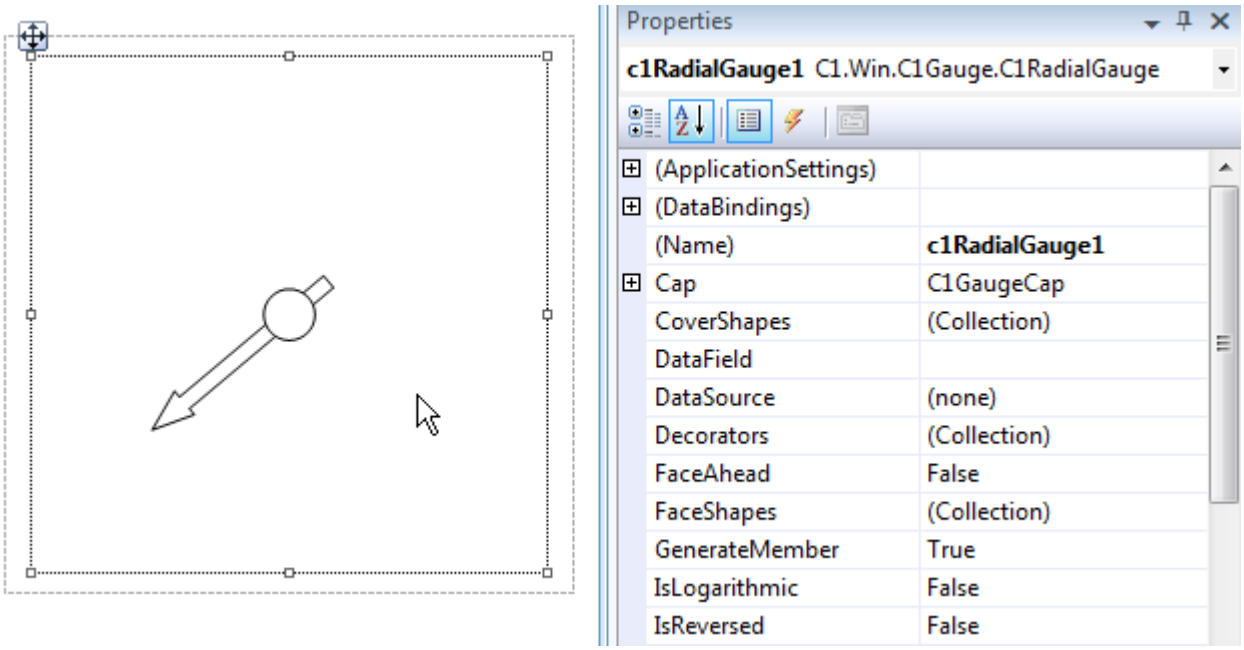
Each task-based help topic also assumes that you have created a new .NET project.

## Editing Gauges at Design Time

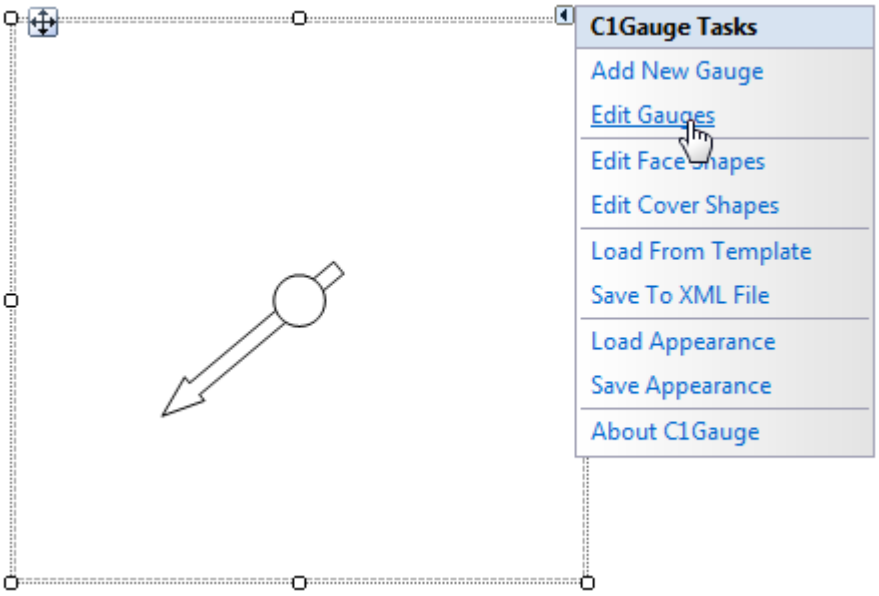
The C1Gauge control is a container for multiple gauges. By default, you start with 1 radial or linear gauge in the container.

To edit the gauges within C1Gauge you can:

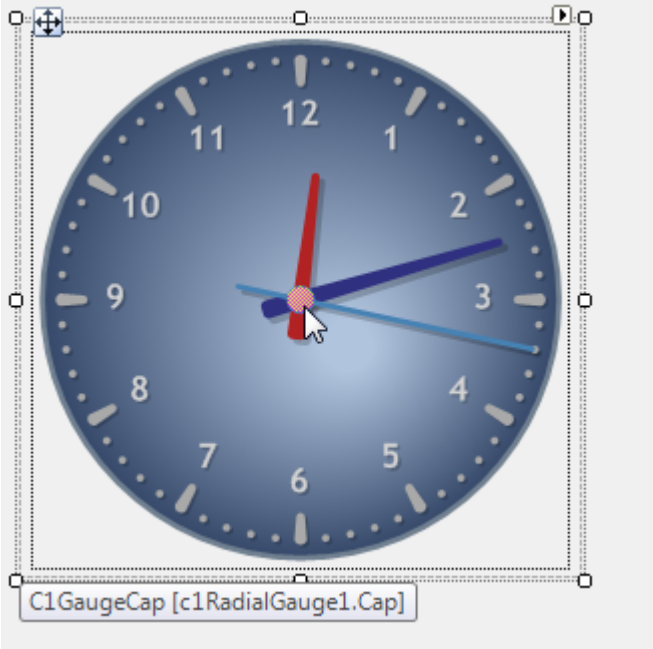
- Click and select the gauge inside the C1Gauge container on the form. Then, the properties window will now reflect the selected gauge.




- Select "Edit Gauges" from the **C1Gauge Tasks** menu. This will open up the properties collection window where you can easily select each inner gauge component to modify its properties.



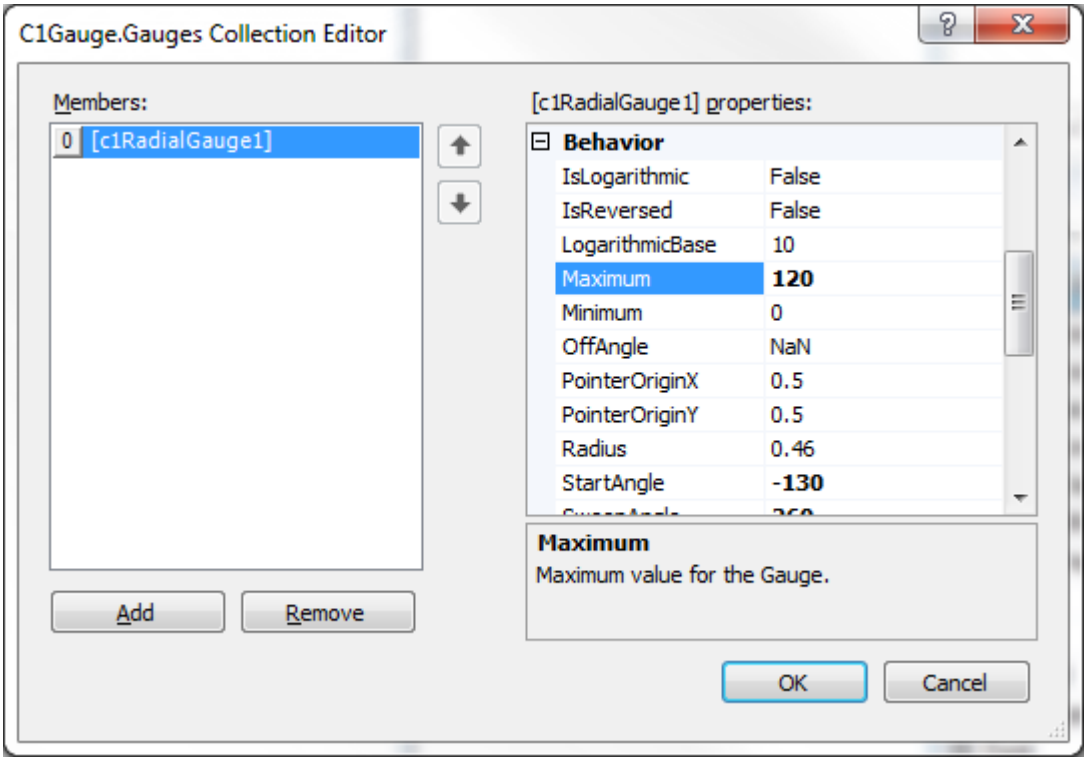

- Double-click an item on the gauge, the cap for example, to open the **Item Editor** and set the item's properties.




## Setting up the Scale

Each gauge inside C1Gauge has its own scale and set of marks and labels. Our empty gauge has none of these yet so let's first declare our maximum and minimum values.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Set the Minimum property to **0** and the Maximum property to **120**. The gauge scale will run from 0 to 120.

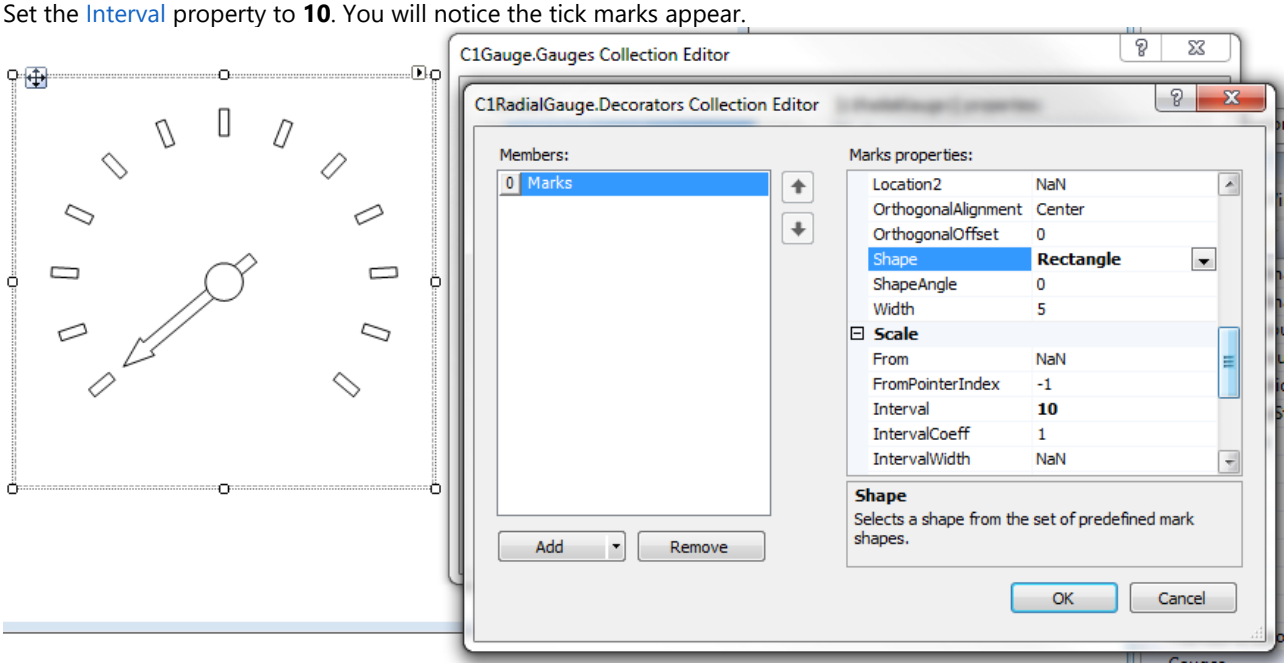



You won't see anything until you add some tick marks or labels. To add these, open the Decorators collection editor.

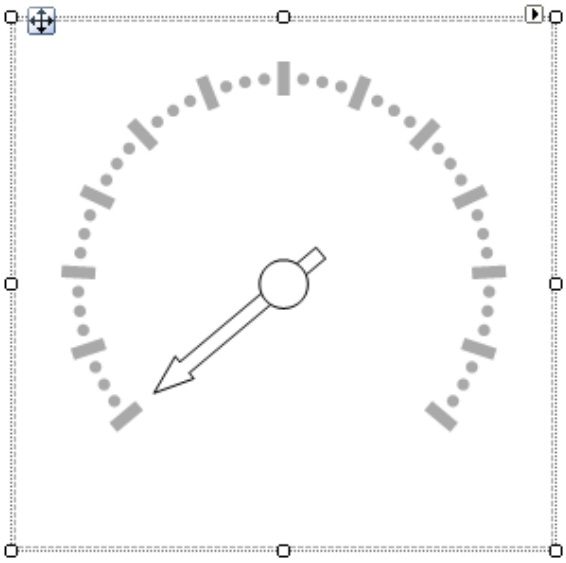
## Adding Tick Marks

In this example, we'll add tick marks to our gauge.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **Decorators**. The **Decorators Collection Editor** opens.
3. From the **Decorators Collection Editor**, drop-down the **Add** button and select C1GaugeMarks. These will be our major tick marks that display at every $10^{th}$ interval.
4. Set the Interval property to **10**. You will notice the tick marks appear.



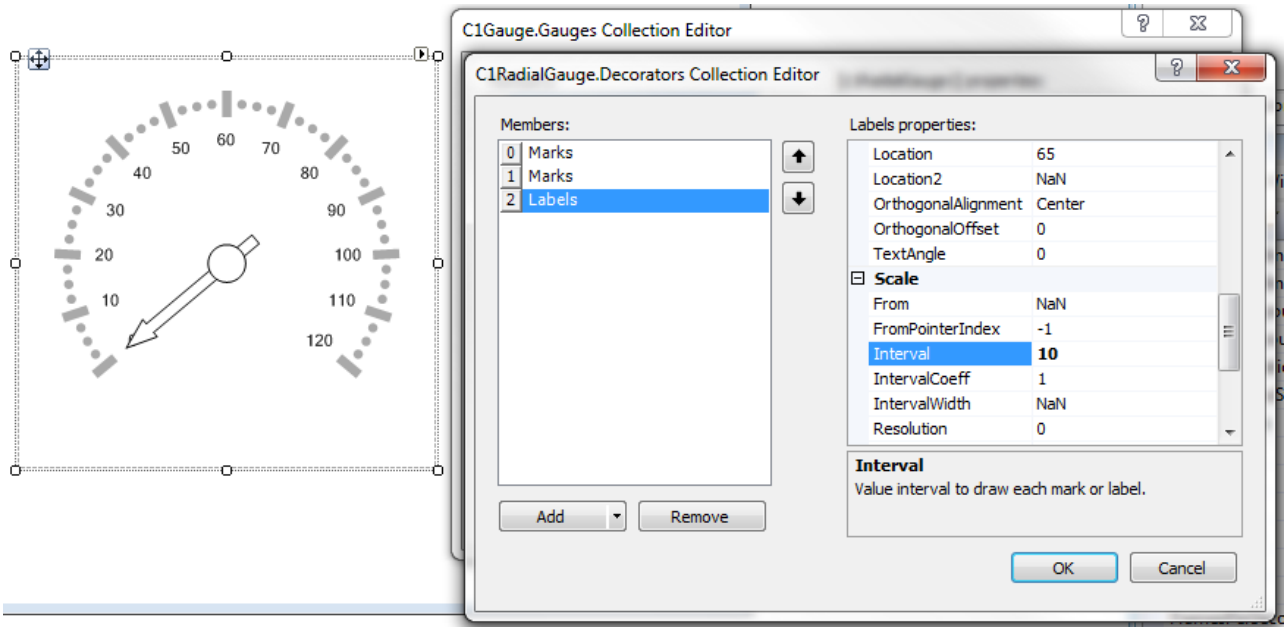
5. By default, the marks have a round shape. Let's make these rectangular by changing the **Shape** property to **Rectangle**.
6. Under **Appearance**, expand **Filling** and set the **Color** property to **DarkGray**. Then remove the border by expanding **Border** and setting the **LineStyle** property to **None**.
7. To add minor tick marks, add another C1GaugeMarks item in the **Decorators Collection Editor**. For these, set the **Interval** to **2.5** so we get three minor ticks between each major mark.
8. We'll keep these marks round in shape, but let's make them shorter by setting the **Length** property to **5** (to match the **Width** property, which is also **5**).
9. Expand **Filling** and set the **Color** property to **DarkGray**. Then remove the border by expanding **Border** and setting the **LineStyle** property to **None**.



## Adding Tick Labels

In this example, we'll add tick labels to the tick marks on our gauge.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **Decorators**. The **Decorators Collection Editor** opens.
3. From the **Decorators Collection Editor**, drop-down the **Add** button and select C1GaugeLabels. These will be the numeric labels that display by each tick-mark.
4. Set the Color property to **DarkGray**.
5. Set the Interval property to **10**. This will display a label at every $10^{th}$ interval from our minimum to our maximum (specified by the minimum and maximum properties set at the gauge level).




6. To customize when the labels start, we can override the minimum and maximum at the Labels level by setting the **From** and **To** properties. For example, let's only display labels on values **20** and higher. Set the From property to **20**. Notice the first two labels are gone.
7. To rotate the labels, set the IsRotated property to **True**. Set the TextAngle property to further customize the rotation of these labels.
8. Set the FontSize property to **10**.
9. Click **OK** to close the **Decorators Collection Editor**.

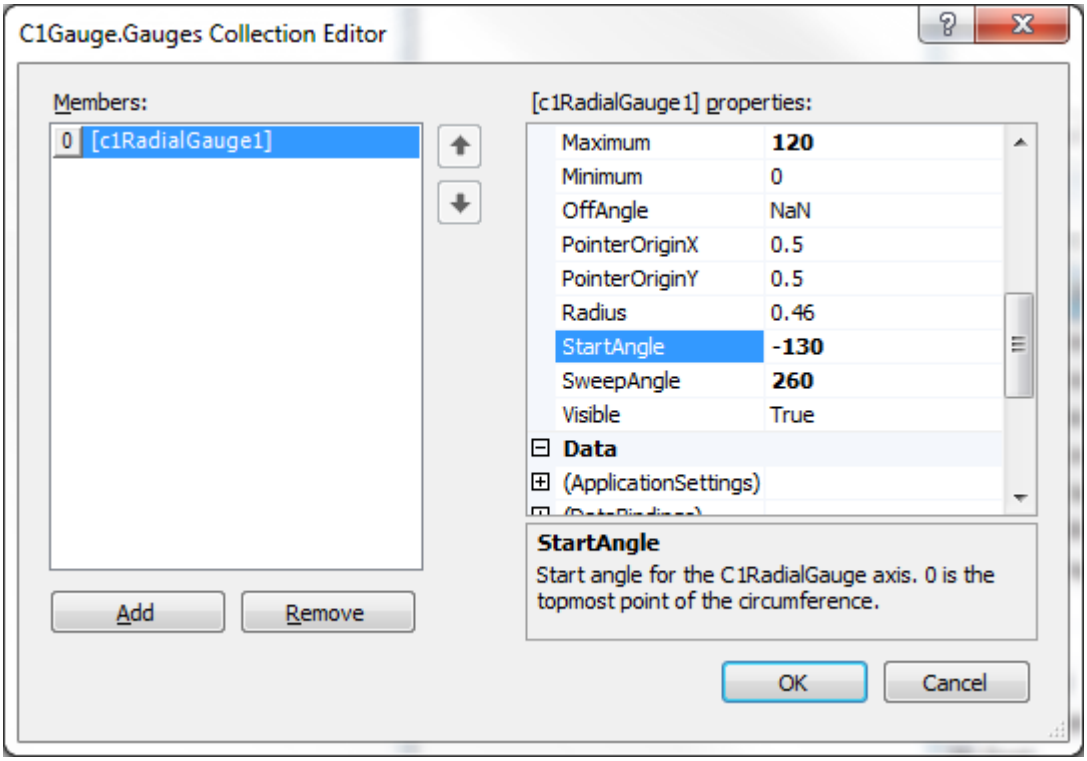
## Adjusting the Starting and Sweep Angles

C1RadialGauge supports various radial shapes including helical, clamshell, and half-circular. We will create these shapes in the next section, but first we can modify the start and sweep angles of the gauge scale to match any of these face shapes.

The StartAngle property defines an angle for the Minimum value. The StartAngle + SweepAngle properties correspond to the Maximum value.

Let's reduce the arc of our scale by setting the **StartAngle** to **-130** and the **SweepAngle** to **260**.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Set the StartAngle property to **-130**.
3. Set the SweepAngle property to **260**.




## Adjusting the Order and Layout of Decorators

By default, you will notice that tick labels are placed inside tick marks. To change the layout and positioning of these elements we have a few key properties available to us. Set these properties on each decorator item, for example, each tick mark and tick label.

| | |
|---|---|
| Alignment: | Choose from 3 simple alignment options: **In**, **Out** and **Center**. |
| Location: | To further change the position of the decorator in relation to the pointer, increase or decrease the value of its Location property. A value of **100** will place the decorator at the far edge, and a lower value places it closer to the origin. |

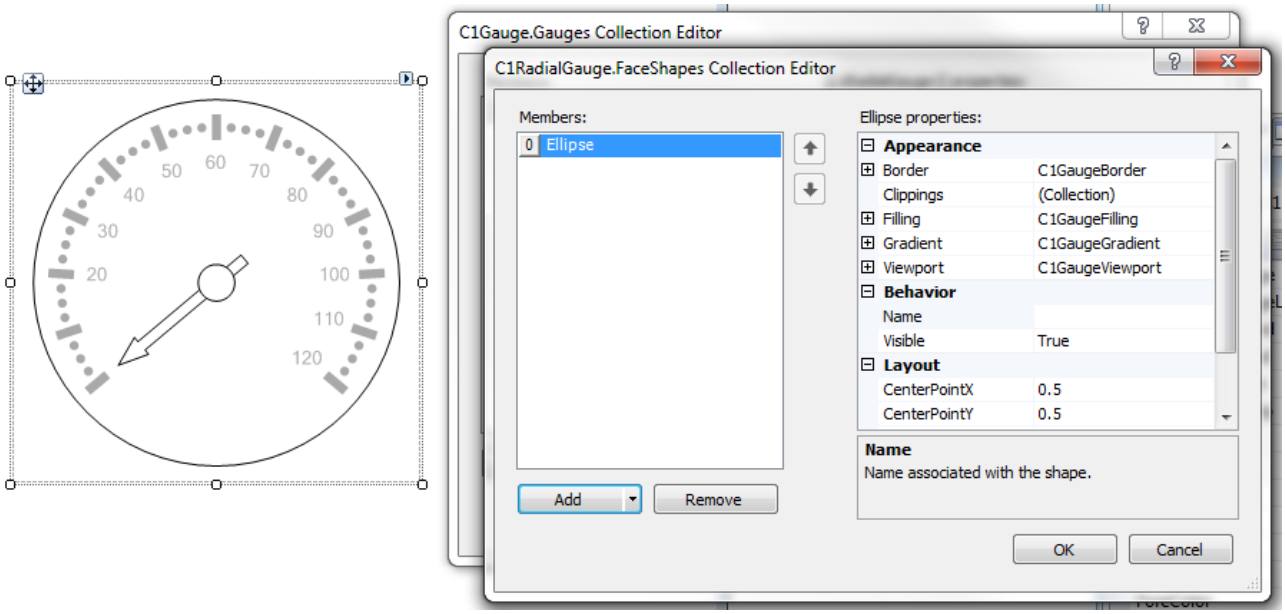
The order in which the items are listed will be the order in which they are placed on the gauge, from top to bottom. To change the order of the items, use the up and down arrows in the **Decorators Collection Editor**, and click **OK**.

## Creating a Face Plate

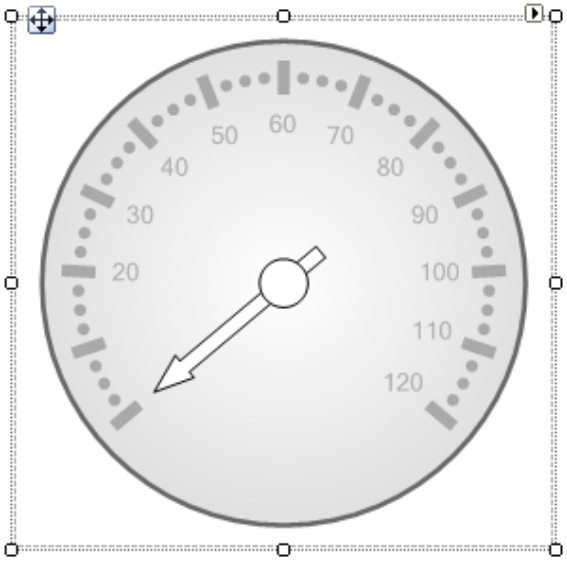
To visually enhance the look of the gauge we need to add a background layer behind our scale, tick-marks and pointer. To do this, we will add shapes to the FaceShapes collection.

From the FaceShapes collection editor, you will notice we can add a variety of shapes including Ellipses, Rectangles, Sectors, Segments or even our own bitmap images.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to the FaceShapes property. The **C1RadialGauge.FaceShapes Collection Editor** opens.
3. Select C1GaugeEllipse from the **Add** drop-down button. We will style this ellipse by giving it a gradient background and thicker border.

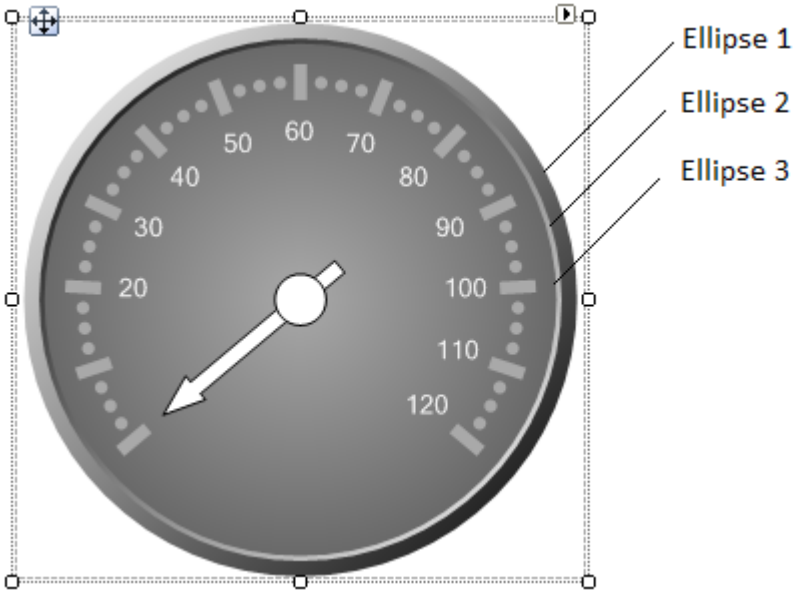



4. Expand **Filling** and set the Color property to **White** and set the Color2 property to **LightGray**.
5. With **Filling** still expanded, set the **BrushType** property to **Gradient**.
6. To modify the gradient direction, expand **Gradient** and set the Direction property to **RadialOuter**. You can also set many more gradient related properties under the **Gradient** node.
7. Expand **Border** and set the Color property to **DimGray** and the Thickness property to **2**.



## Creating a More Complex Face

The Border property of all **FaceShapes** is nice and simple. We can create a more professional-looking beveled edge to our gauge by adding a couple shapes on top of one another and using gradients for a lighting effect. For this example we will have a total of three **Ellipses** in our **FaceShapes** collection.




1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the ellipsis button next to the FaceShapes property. The **C1RadialGauge.FaceShapes Collection Editor** opens.
3. Click **Add** and select C1GaugeEllipse. The first ellipse will be the bottom-most.
4. Expand **Filling** and set the Color property to **White**, Color2 property to **Black**, and BrushType to **Gradient**. We need to make the bottom-most ellipse a bit bigger so we can see it when others are on top. To do this, set the Width and Height properties to **-1.08**. Note that a value of **-1** means 100% of the region. Values less than **-1** will increase the size proportionally. Positive values denote explicit sizes in pixels.
5. Expand **Border** and set the LineStyle property to **None**.
6. Click **Add** and select C1GaugeEllipse. The second ellipse will be slightly smaller than the bottom.
7. Set its Width and Height values to **-1.02**.
8. Expand **Filling** and set the Color property to **Black**, Color2 property to **White**, and BrushType to **Gradient**.
9. Expand **Border** and set the LineStyle property to **None**.
10. Click **Add** and select C1GaugeEllipse. The third ellipse will act as our face plate directly beneath the scale and pointer. We will keep its **Width** and **Height** at **-1** (the default of 100%).
11. Set the Color property to **DarkGray**, Color2 property to **DimGray** and BrushType to **Gradient**.
12. Expand **Gradient** and set Direction to **RadialInner**.
13. Expand **Border** and set the LineStyle property to **None**.

## Customizing the Pointer and Cap

The pointer and cap objects consist of many styling properties. To modify these properties expand the Cap or Pointer property nodes in the **C1Gauge.Gauges Collection Editor** on the selected C1Gauge. With the pointer you can customize the filling, border and gradient properties just like with all shape elements in C1Gauge. You can also choose a predefined shape for the pointer and further customize the shape in the **CustomShape** properties.


## Styling the Pointer

To style the pointer, follow these steps:

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Expand the **Pointer** node, and set the following properties:
    1. Expand **Border** and set the LineStyle property to **None**.
    2. Expand **Filling** and set BrushType to **Gradient**, Color to **Orange**, and Color2 to **LightGoldenrodYellow**.
    3. Expand Shadow and set **Visible** to **True**.
    4. Set Shape to **Thumb**.

Notice all of the predefined shapes under the Shape property. You can also customize these shapes under the **CustomShape** properties.

## Styling the Cap

To style the cap, follow these steps:

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Expand the **Cap** node, and set the following properties:
    1. Set Radius to **15**.
    2. Expand **Filling** and set the Color property to **White**.
    3. Expand **Filling** and set the Color2 to **Black**.
    4. Expand **Filling** and set the BrushType to **Gradient**.
    5. Expand Shadow and set **Visible** to **True**.

C1Gauge also includes a **MoreCircles** collection for the cap object. This allows you to add more circle shapes to your cap design, each with the same set of styling properties.

## Displaying the Pointer on Top of the Cap

By default, the pointer is underneath the cap. To switch this, simply set the BehindPointers property to **True**.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Expand the **Cap** node, and set the BehindPointers property to **True**.

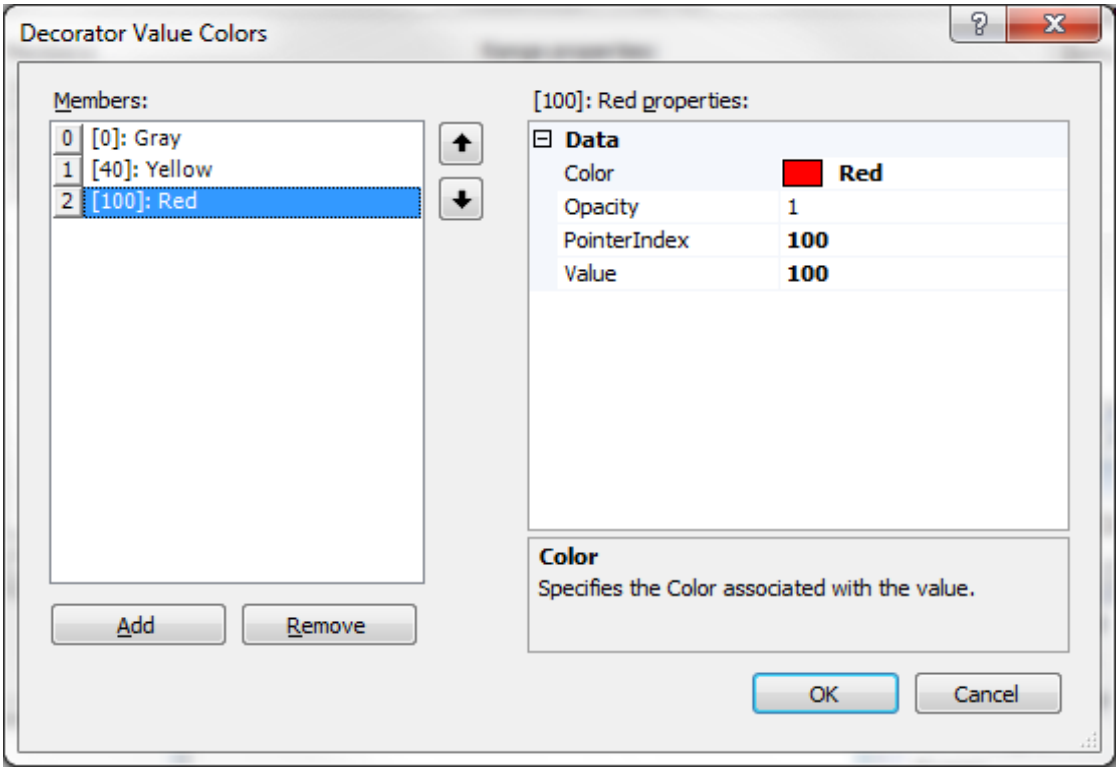
## Adding Ranges

Ranges are bands of color that denote certain thresholds along the gauge scale. You can add ranges to C1Gauge through the **Decorators** collection, along with tick marks and labels.

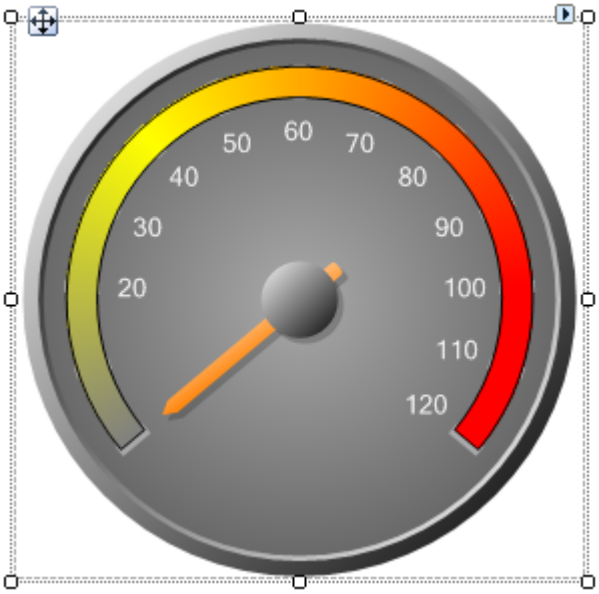
1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **Decorators**. The **Decorators Collection Editor** opens.
3. Click the drop-down arrow next to **Add** and select C1GaugeRange. Just like with our **Marks** and **Labels**, we can position the range by setting the Alignment and/or Location properties.
4. Set the Location property to **85** so that it aligns with our tick marks.


Set the From and To properties to specify the limits of this range. Since we want this range to go from our minimum to our maximum we will leave these as default (NaN). To mark various color-based thresholds you could add multiple ranges, each with a different Filling color. However, C1Gauge provides a convenient way to map colors to values using the **ValueColors** collection.

5. You may have noticed that all **Decorator** items (including tick marks) have a **ValueColors** collection. Select the range we just added and click the **ellipsis** button next to **ValueColors** to open the **Decorator Value Colors** window.
6. Click **Add** three times to add three colors: gray, yellow and red. For each color we specify the color property, and the corresponding value. The value set is the minimum value for the range (i.e., gray: 0, yellow: 40, red: 100). For each color we also select the pointer which gives the value.
    1. Select the first member 0, click the drop-down arrow next to **Color**, and select **Gray**. Set the PointerIndex and Value properties to **0**.
    2. Select the next member 1, click the drop-down arrow next to **Color**, and select **Yellow**. Set the PointerIndex and Value properties to **40**.
    3. Select the last member 2, click the drop-down arrow next to **Color**, and select **Red**. Set the PointerIndex and Value properties to **100**.

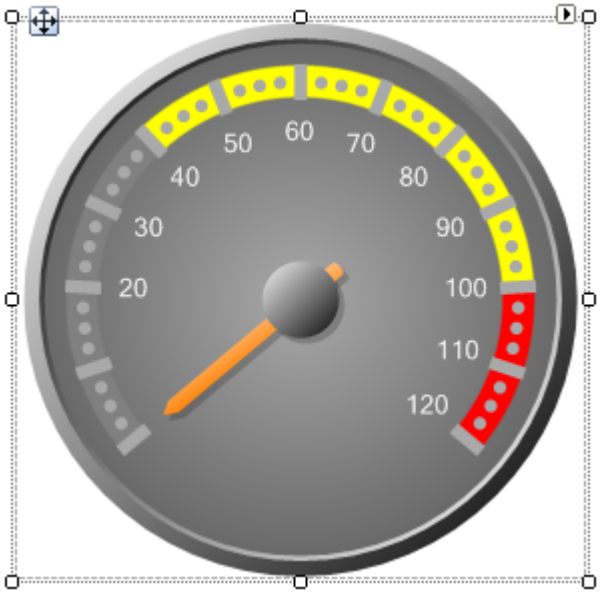



7. Click **OK** to close the **Decorator Value Colors** window.




You will notice the colors blend together across the range. We could, however, remove the blending effect of the colors by setting the ValueColorFalloff property to **None** in the **Decorators Collection Editor**.

To move the range below the tick marks, use the up/down arrows to move it to the $0^{th}$ member. In this example, we also removed the border of the range for a better look. Click **OK** to close the **Decorators Collection Editor**.




## Enhancing Your Ranges

There are several properties you can use to further enhance your ranges.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **Decorators**. The **Decorators Collection Editor** opens.
3. Select the desired **Range** and set any of the properties described below.

To create a non-linear range, set the Width2 property to a different value than Width.

To enhance the graphic rendering of ranges without borders, set the AntiAliasing to a higher quality.

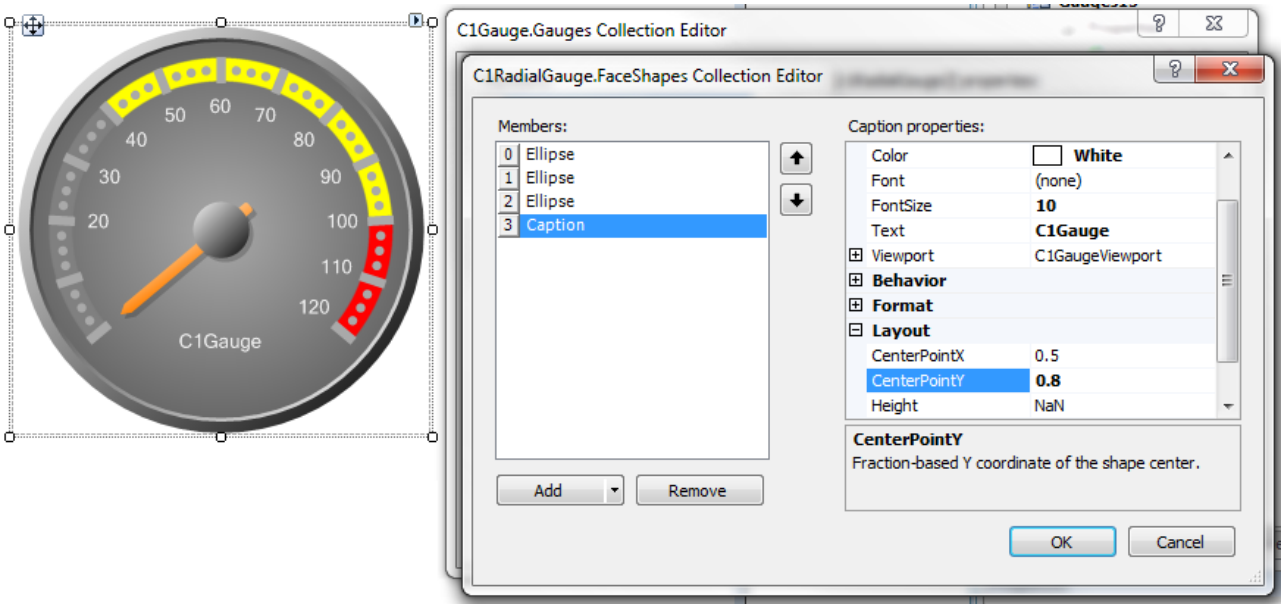
To create a moveable range that is bound to the pointer value, set the ToPointerIndex property to the index of the pointer (a high value such as 100 will bind to the main pointer at all times). You can also bind the low end of a range to a pointer using the FromPointerIndex property.

To use a range as the gauge value indicator (such as with a thermometer gauge), set the ToPointerIndex property to the index of your pointer and then set the pointer's Visible property to **False**.

## Adding Captions

It's common to display text on a gauge, such as units of measure or labels. You can add any number of captions through the **FaceShapes** or **CoverShapes** collections.

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **FaceShapes**. The **FaceShapes Collection Editor** opens.
3. Select C1GaugeCaption from the **Add** drop-down button.
4. Set the Text property of the caption to "C1Gauge." You will notice the caption appears behind the **Cap**. If you added the caption to the **CoverShapes** collection, it would display on top.
5. Set the Color property to **White** and the FontSize to **10**.
6. If you don't want the caption in the middle of the gauge, move it down by setting the CenterPointY property. By default, this is 0.5, which means the caption appears 50% of the way down the gauge from the top. Set CenterPointY to 0.8 to move it down.




No matter what size the gauge is scaled at, this Caption will always appear at that expected location. Here, I've also set the **FontSize** to **10** and the **FontColor** to **White**.

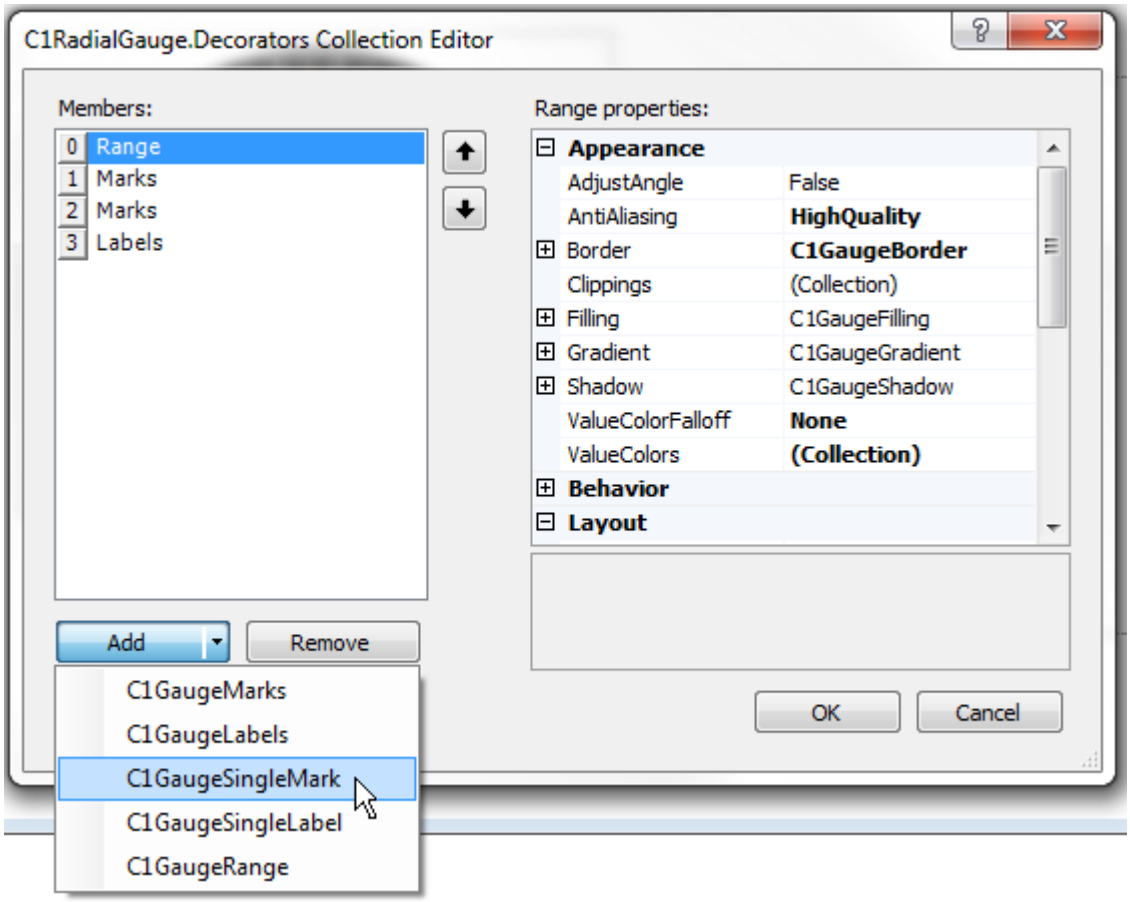
## Creating a State or Numeric Indicator

There is no special support for state indicators in the first version of **Gauges for WinForms**, however we can achieve similar effects using bound labels and markers.

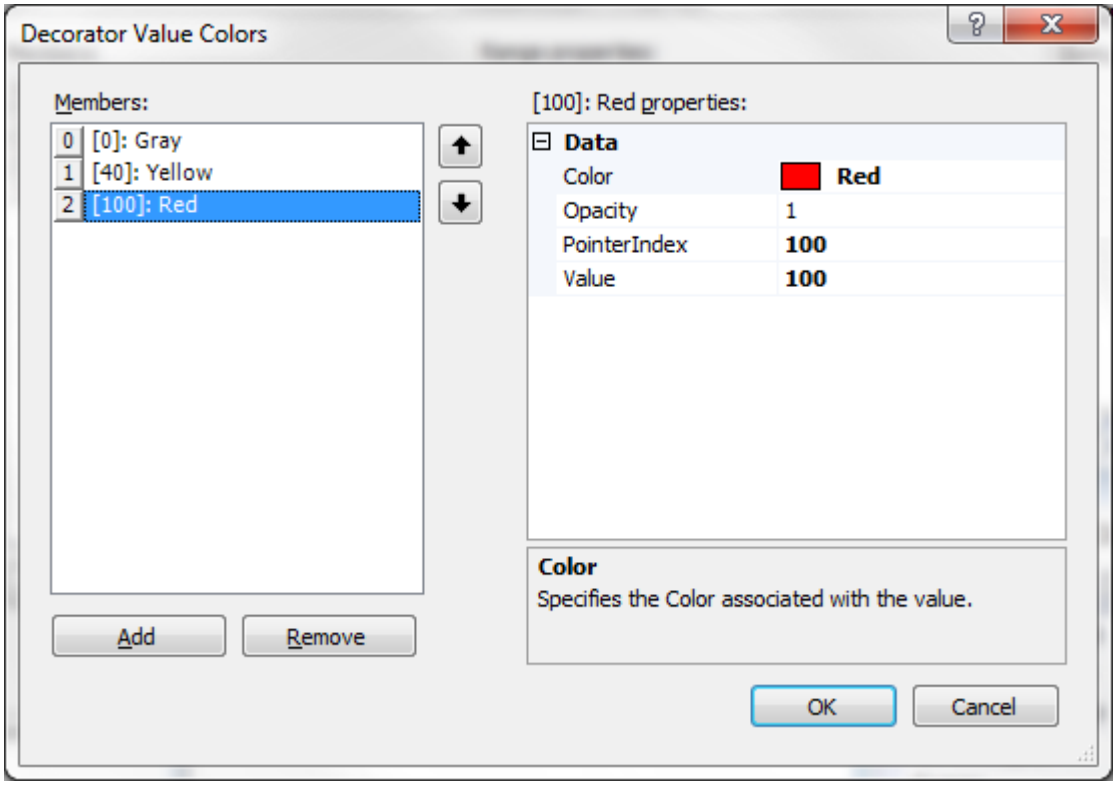
## State Indicator

To create a state indicator, follow these steps:

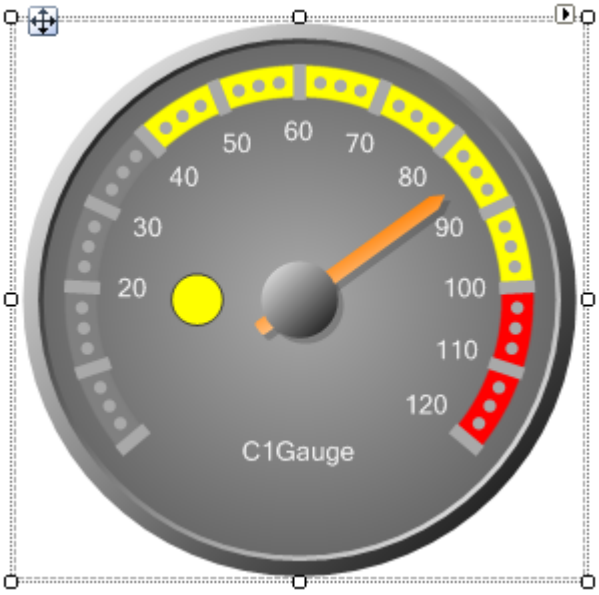
1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **Decorators**. The **Decorators Collection Editor** opens.
3. Click the **Add** drop-down arrow and select C1GaugeSingleMark.




4. To make the **SingleMark** visible, set the PointerIndex property to **1**. You will notice it appears on the minimum value of your scale.
5. To position the **SingleMark**, set the Angle property to **270**. Then set the Location property to **40**. You will notice now it appears to the left of the pointer (around value 20).
6. Also, set the Width value to **15** and Shape to **Round**.
7. Expand the **Border** node and set the **Color** property to **DarkGray**.
8. We need this indicator to show the same color as the value of the gauge, as determined by our Range we set up in the Adding Ranges topic. To do this, we need to add the same color to value thresholds in the **ValueColors** collection for our **SingleMark**. Click the **ellipsis** button next to **ValueColors**. In the **Decorator Value Colors** window, add 3 members: **Gray**, **Yellow** and **Red** with the corresponding **Values** of **0**, **40** and **100** respectively. This is just like we did for the range.
    1. Select the first member 0, click the drop-down arrow next to **Color**, and select **Gray**. Set the PointerIndex and Value properties to **0**.
    2. Select the next member 1, click the drop-down arrow next to **Color**, and select **Yellow**. Set the PointerIndex and Value properties to **40**.
    3. Select the last member 2, click the drop-down arrow next to **Color**, and select **Red**. Set the PointerIndex and Value properties to **100** and click **OK** to close the window.




9. Finally, set the ValueColorFalloff property to **None** so that it matches the same colors as our range (without gradient blending).



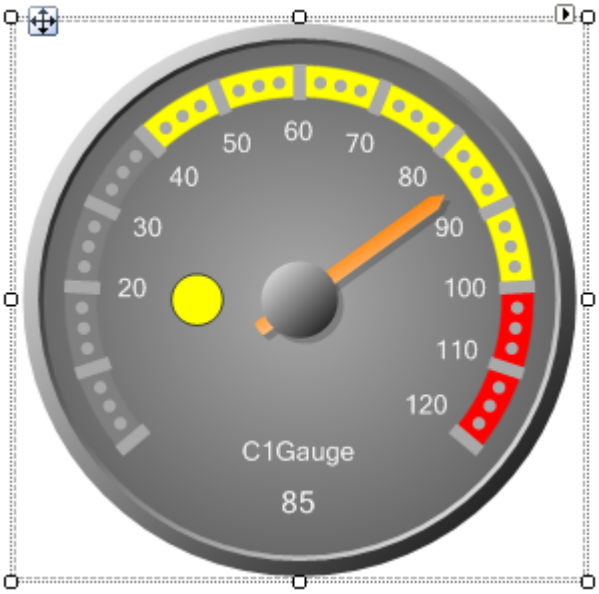

Notice that since we have declared the value colors separately for the **SingleMark**, we do not need the range anymore on our gauge.

## Creating a Numeric Indicator

To add a numeric indicator, follow these steps:

1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **Decorators**. The **Decorators Collection Editor** opens.

3. Click the **Add** drop-down arrow and select C1GaugeSingleLabel.
4. To bind the C1GaugeSingleLabel to the pointer value, simply set the PointerIndex to **1**.
5. To position the C1GaugeSingleLabel, set its Angle property to **180** and Location to **80**. This will position the label centered below our caption. You can increase the FontSize to 12.



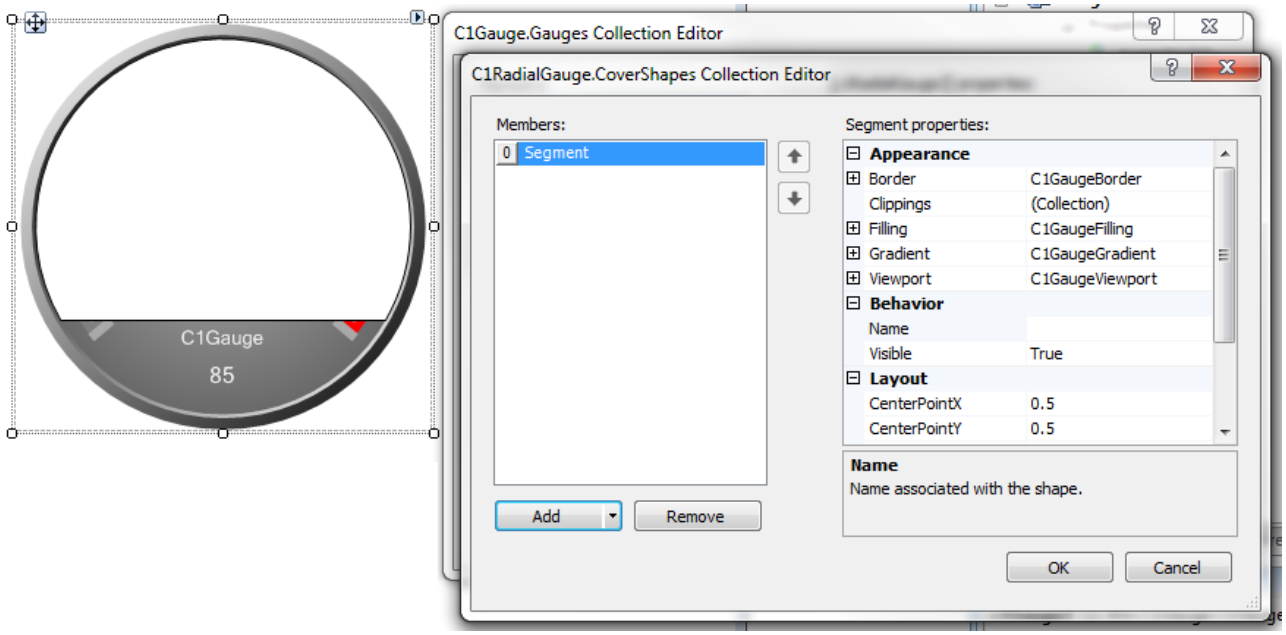

Notice, we can also apply Value Colors to this **SingleLabel**, meaning the label could be Yellow for the given value above. You would just apply the same set of Value Colors as we did for the State Indicator.

## Adding a Glass Effect

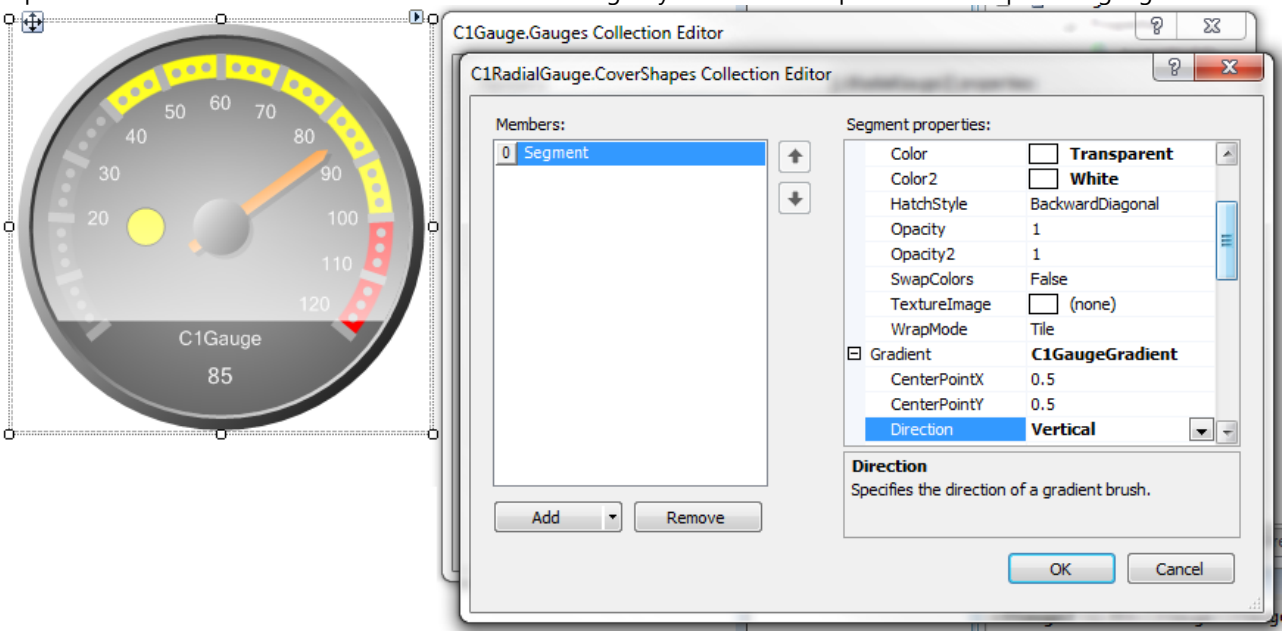
In addition to the **FaceShapes** collection, each gauge in C1Gauge has a **CoverShapes** collection. The only difference is that the **CoverShapes** items draw on top of all gauge elements.

To create a glassing effect, we will add a semi-transparent segment.

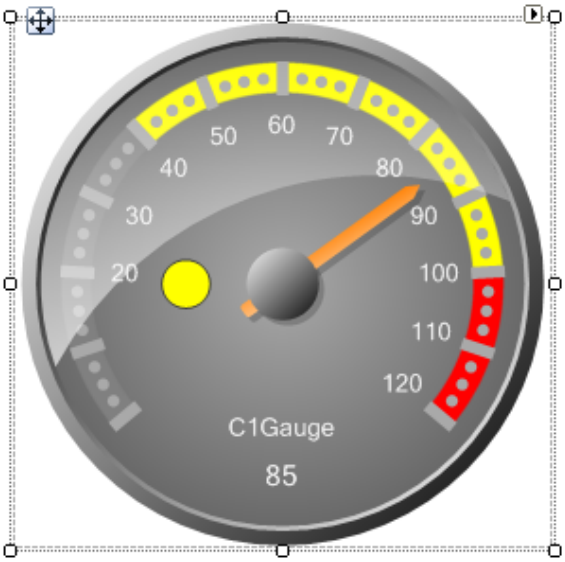
1. Click the **ellipsis** button next to the Gauges property in the Properties window. The **C1Gauge.Gauges Collection Editor** dialog box opens.
2. Click the **ellipsis** button next to **CoverShapes**. The **CoverShapes Collection Editor** opens.
3. Click the **Add** drop-down arrow and select C1GaugeSegment.




4. Remove the border of the **Segment** by expanding **Border** and setting LineStyle to **None**.
5. Expand **Filling** and set the BrushType property to **Gradient**. Set Color to **Transparent** and Color2 to **White**.
6. Expand **Gradient** and set Direction to **Vertical**. This will give you a semi-transparent fill on top of the gauge.




7. Finally, to create an arc to simulate a real glass-looking gauge, we need to set a few angle properties. Set the StartAngle to **-110**, the SweepAngle to **180** and set the InnerRadius property to **150**. This will give us the desired glassing effect.




To achieve glassing effects for non-circular shapes, we can take advantage of the **Clippings** collections.